| | |
|---|---|
| **Titre:** Title: | Comparaison quantitative des algorithmes de simplification des maillages triangulaires |
| **Auteur:** Author: | Mikhail Zoline |
| **Date:** | 2005 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Zoline, M. (2005). Comparaison quantitative des algorithmes de simplification des maillages triangulaires [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7694/ |



| | |
|---|---|
| **URL de PolyPublie:** PolyPublie URL: | https://publications.polymtl.ca/7694/ |
| **Directeurs de recherche:** Advisors: | |
| **Programme:** Program: | Non spécifié |

UNIVERSITÉ DE MONTRÉAL

# COMPARAISON QUANTITATIVE DES ALGORITHMES DE SIMPLIFICATION DES MAILLAGES TRIANGULAIRES

MIKHAIL ZOLINE
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
AOÛT 2005

*Your file Votre référence*
*ISBN: 978-0-494-16870-7*
*Our file Notre référence*
*ISBN: 978-0-494-16870-7*

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

# COMPARAISON QUANTITATIVE DES ALGORITHMES DE SIMPLIFICATION DES MAILLAGES TRIANGULAIRES

présenté par : ZOLINE, Mikhail

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

Mme CHERIET Farida, Ph.D., présidente

M. GUIBAULT François, Ph.D., membre et directeur de recherche

M. GRANGER Louis, M.Sc., membre

# REMERCIEMENTS

Je tiens à remercier sincèrement mon directeur de recherche, monsieur François Guibault pour tout le support académique et moral qu'il m'a apporté dans la réalisation de ce projet, ainsi que pour son analyse scientifique afin de me guider dans toutes les étapes que j'ai eu à franchir au cours de mes études supérieures.

Je le remercie également pour le support financier qui me fut octroyé, me permettant de me consacrer entièrement à ce projet, sans lequel soutien la réalisation aurait été impossible.

Un merci particulier à ma conjointe Linda Clément pour son aide précieuse lors de la rédaction de ce mémoire.

# RÉSUMÉ

Dans le domaine de la génération des maillages, il existe de nos jours plusieurs algorithmes de simplification ou décimation de maillages triangulaires. Nous implémenterons sur la même plate-forme trois de ces algorithmes basés sur les opérations de réduction des arêtes et séparation des sommets. Afin de répondre aux questions relatives au choix de l'algorithme le plus robuste et efficace, les résultats de ces algorithmes seront comparés numériquement selon le niveau d'approximation du maillage initial, d'abord visuellement, puis en termes de temps d'exécution et de consommation de mémoire. L'emphase sera principalement mise sur les aspects quantitatifs de la comparaison.

La comparaison visuelle sera donc effectuée selon le niveau de détail de chacun des algorithmes, pour un taux de simplification de 10 % à 95 % de réduction du nombre de triangles du maillage initial. Les maillages qui seront utilisés dans notre travail seront obtenus à l'aide des fonctions de rappel de l'interface NURBS d'OpenGL. Cette façon de faire nous donnera la possibilité de procéder à la décimation des surfaces NURBS représentées par un grand nombre de points de contrôle, ce qui s'avérait impossible avec les seules fonctions modifiant les propriétés des surfaces NURBS d'OpenGL.

**Mots clés**

Comparaison quantitative des algorithmes de simplification, décimation de maillages triangulaires, métrique d'erreur, niveau de détail, surfaces NURBS.

# ABSTRACT

Many different algorithms for decimation or for simplification of triangular meshes have been proposed in the last few years. We will implement on the same platform, three of these algorithms, based on iterative vertex pair contractions. In order to determine the best alternative relative to the choice of the most efficient and the most robust algorithm, we will compare the results of these algorithms numerically, i.e. in terms of approximation error introduced in the simplification process, visually at first, and then in terms of execution time and memory consumption.

We will focus primarily on the quantitative aspects of our comparisons. The visual comparison of each algorithm will be carried out for ten different levels of details for one simplification rate from 10% to 95% of triangles reduction. All meshes that will be used in this process have been obtained via callback functions of the OpenGL NURBS Interface. Using this procedure enabled us to simplify NURBS surfaces represented by large number of control points, which appeared to be an impossible task via OpenGL NURBS Interface.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# LISTE DES ANNEXES

# CHAPITRE 1

# INTRODUCTION

Si une image vaut mille mots, peut-on dire que, dans le monde des communications visuelles d'aujourd'hui, une image numérique vaut un million de mots?

De nos jours, les progrès scientifiques nous permettent d'accroître les taux de réussite dans bien des domaines; les domaines des génies aérospatial, énergétique et industriel en sont quelques exemples. Afin de répondre rapidement et efficacement aux demandes croissantes en besoin d'analyse dans ces domaines, les simulations numériques (qui mettent en oeuvre des modèles théoriques de phénomènes réels) sont grandement utilisées. Ces simulations sont généralement plus économiques et plus souples que les expériences réelles même si des validations sont toujours nécessaires pour garantir leur pertinence.

Par exemple, en aéronautique un très grand nombre de simulations sont utilisées lors du design d'un nouveau modèle, entre autres afin d'en accroître la performance tout en calculant la robustesse des pièces mécaniques (ailes, fuselage, turbines de moteurs, etc.) mais également pour s'assurer de la sécurité des équipements. Dans le domaine du génie civil, on a recours à la simulation numérique pour les gratte-ciel avant leur construction, soit à l'étape du design, afin d'augmenter la robustesse et la sécurité dans toutes les conditions environnementales et climatiques possibles à l'endroit où la construction aura lieu. Dans le domaine énergétique, lors de la construction de grands barrages par exemple, la simulation numérique permettra d'établir avec précision la forme des turbines et des pièces d'écoulement à utiliser, ce qui sera un gage de performance et de sécurité avant même la construction et la mise en fonction du barrage

## 1.1 Motivation

Dans le cadre de notre Projet de fin d'études, nous avons été appelé à travailler à la mise au point et à la maintenance d'un logiciel de visualisation, soit Topovisu.

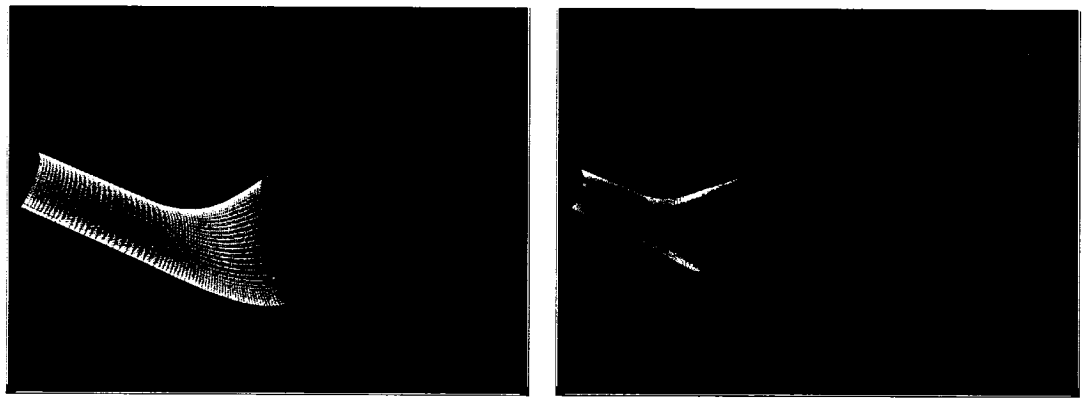
Le logiciel Topovisu a été conçu afin de permettre la visualisation des données à n'importe quel moment de la démarche du design. Ce logiciel, considéré comme un outil de support au développement des applications, est utilisé lors de la validation du modèle géométrique d'une simulation numérique. Le format utilisé par ce logiciel, c'est-à-dire le format « .pie » prévoit l'utilisation des surfaces NURBS afin de décrire un modèle.

Lors du processus de design assisté par ordinateur, les utilisateurs de Topovisu doivent, afin de conserver la précision numérique du modèle, utiliser plusieurs milliers de points de contrôle.

Dans la version antérieure de Topovisu, l'affichage des surfaces avait été entièrement conçu via OpenGL, plus particulièrement via l'interface NURBS.

En effet, lors de l'affichage d'une surface NURBS, l'interface NURBS d'OpenGL permettait plusieurs possibilités de modification des propriétés des surfaces restituées. Cependant, dans le cas où la surface était représentée par un très grand nombre de points de contrôle, la modification des propriétés s'avérait non opérationnelle et ce, étant donné qu'OpenGL subdivise la surface NURBS en « carreaux de Bézier ». À ce stade, les « carreaux de Bézier » passent à l'évaluateur de bas niveau d'OpenGL, qui effectue une grande partie de la restitution finale.

Lorsqu'un « carreau de Bézier » représente une infime partie de la surface, l'évaluateur de bas niveau d'OpenGL est alors forcé de le restituer avec tous les points de contrôle dudit carreau ce qui engendre un très grand nombre de points de restitution pour la surface complète.



(a) (b)

FIGURE 1.1 – Surface NURBS représentée par environ 4,000 points de contrôle.

- *La figure 1.1a montre une surface rognée et dessinée en mode « fil de fer ».*
- *La figure 1.1b montre la surface illustrée à la figure 1.1(a) non rognée et représentée par son nuage de points de contrôle.*

La précision de la représentation numérique des surfaces dépend entre autres du nombre de points de contrôle; un trop grand nombre de ces points de contrôle provoque un affichage lent et surchargé. La précision de l'affichage dans un tel cas n'est pas affectée et demeure très bonne mais la performance (rapidité), quant à elle, est affectée.

Lors de l'affichage d'un volume représenté par plusieurs dizaines de surfaces NURBS contenant un très grand nombre de points de contrôle, les performances de la majorité des postes de travail sont insuffisantes pour permettre l'affichage interactif des volumes en mode « temps réel ».

Pour arriver à simplifier les surfaces affichées par l'interface NURBS d'OpenGL, nous avons profité de la possibilité que nous offre cette interface afin d'avoir accès aux informations sur les sommets des objets géométriques résultant de la fragmentation des courbes et surfaces (interface « rappel » de NURBS). En utilisant l'information géométrique récupérée, nous pouvons construire un maillage afin de pouvoir utiliser les algorithmes de simplification de maillages.

Le présent travail aura donc pour but de comparer, sur la base de statistiques des erreurs numériques, trois algorithmes de décimation de surfaces. Ceci aura pour effet de permettre le choix du meilleur algorithme afin d'améliorer les performances d'affichage des surfaces NURBS d'OpenGL contenant un très grand nombre de points de contrôle.

## 1.2 Contribution

Le but premier de la présente recherche est de choisir l'algorithme de simplification de maillage triangulaire ayant le meilleur rapport coût/précision afin de l'intégrer dans Topovisu pour améliorer le temps d'affichage des modèles composés de surfaces contenant un très grand nombre de points de contrôle.

En tout, trois (3) algorithmes de simplification seront implémentés sur la même plate-forme et seront comparés selon quatre (4) critères. Nous effectuerons ces tests dans le but de comparer l'efficacité de l'algorithme très répandu de M. Garland et al., dont la première version date de 1997, avec un nouvel algorithme soit celui de M. Hussain, Okada et Niijima datant de 2001 et avec l'algorithme de S.-J. Kim et al., également nouveau, datant de 2002. Nous planifions améliorer ce dernier selon les deux aspects suivants:

1. Définition complète de l'algorithme dans le cas où celui-ci n'est pas bien documenté ou défini dans les articles de référence et;
2. Modification de l'opérateur de courbure principale utilisé par cet algorithme afin d'améliorer le temps d'exécution tout en conservant la précision de l'algorithme.

En effet, M. Hussain et ses collaborateurs ont publié un nouvel algorithme de décimation de maillage basé sur les mesures de déviation géométrique et l'importance visuelle des sommets afin de prévenir la décimation des parties comportant un haut niveau de détails.

En plus, cet algorithme laisse présager une excellente efficacité en terme de consommation de mémoire comparativement à celui de M. Garland.

Un atout à ne pas négliger de ce nouvel algorithme est sa facilité d'implémentation. Le présent travail aura donc pour but de comparer les trois algorithmes suivants :

1. Simplification des surfaces via l'erreur quadratique avec pondération par la superficie des triangles (M. Garland, 1999)
2. Simplification des surfaces via l'erreur géométrique (M. Hussain, Y. Okada, K. Niijima, 2004)
3. Simplification des surfaces via l'erreur de courbure discrète (S.-J. Kim et al., 2002)

Nous procéderons également à diverses analyses des algorithmes dont les résultats seront évalués en fonction des **quatre** critères suivants :

1. **Le temps d'exécution** de chaque algorithme sera mesuré pour un taux fixe de réduction de triangles, permettant de comparer les performances.
2. **La comparaison numérique** utilisera une centaine de modèles initiaux qui seront simplifiés pour un taux fixe de réduction de triangles. Les modèles simplifiés seront sauvegardés sous forme de fichiers ASCII. La comparaison numérique par rapport au maillage initial sera effectuée à l'aide du logiciel MESH pour chacun des algorithmes. Ce test nous permettra d'évaluer la précision numérique de chacun des algorithmes.
3. **La consommation de mémoire** sera évaluée pour chacun des algorithmes selon les expressions formelles de la quantité de mémoire requise, spécifiquement pour l'exécution de chacun.

4. **La comparaison visuelle,** après une simplification pour un taux de réduction fixe, sera effectuée sur plusieurs modèles initiaux comportant un niveau de détails élevé. Ce test nous permettra d'évaluer et de comparer visuellement les performances des algorithmes dans le but de conserver le niveau de détails le plus élevé possible.

Les surfaces utilisées dans le présent travail seront extraites de modèles réels conçus pour le domaine hydroélectrique. Ces modèles seront transformés de la description NURBS en format polygonal. Deux caractéristiques communes à toutes ces surfaces testées seront :

1. De comprendre un très grand nombre de points de contrôle
2. D'y retrouver la présence de courbes de rognage

En tout, nous planifions avoir recours à un échantillonnage d'une centaine de surfaces afin d'effectuer l'analyse statistique.
Lors de la comparaison numérique, nous utiliserons la méthode d'évaluation d'erreurs selon la distance de Hausdorff. *(Les métriques d'erreurs seront décrites à la section 2.3.2)*

Les résultats de ce travail pourront être utilisés à titre de référence dans :

1. Le choix d'un algorithme de simplification des maillages triangulaires ou ;
2. Le choix d'un algorithme lors de la simplification d'une surface NURBS d'OpenGL comportant un très grand nombre de points de contrôle et donc de points de restitution.

## 1.3 Structure du document

En premier lieu, nous débuterons le prochain chapitre par la présentation des notions principales nécessaires à la simplification des surfaces et poursuivrons par un survol des travaux précédents au chapitre 3. Le chapitre 4 présentera en détail les trois algorithmes utilisés. Au chapitre 5, nous présenterons la méthodologie de comparaison statistique des algorithmes. Le chapitre 6 sera consacré à une discussion des résultats. Finalement, le chapitre 7 nous permettra de conclure sur le travail effectué.

# CHAPITRE 2

# CONTEXTE DU TRAVAIL

Ce chapitre a pour but de présenter le contexte du présent travail et de résumer les notions principales qui seront utilisées dans les chapitres ultérieurs. Nous présenterons d'abord les modes de représentation des surfaces paramétriques et polygonales.
Ensuite, nous passerons à la méthodologie générale de simplification des surfaces polygonales et à l'évaluation de la qualité de l'approximation des surfaces. Enfin, nous nous pencherons sur les notions de métriques reliées à l'évaluation de la qualité d'approximation des surfaces.

## 2.1 Représentation des surfaces

De nos jours il existe plusieurs formes de représentations de surfaces. Les deux formes les plus répandues sont la représentation paramétrique et la représentation polygonale. Cependant, d'autres alternatives existent, telles les surfaces de subdivision et les surfaces implicites employées par les algorithmes des modèles volumétriques (*comme Marching Cubes*) (Lorensen, W.E., Cline, H.E., 1997). Toutes les représentations de surfaces comportent leurs avantages et inconvénients ; elles sont employées afin de répondre à certains besoins spécifiques. Par exemple, les surfaces de subdivision sont largement employées dans la modélisation et l'animation 3D (SIGGRAPH 2000, Course notes) tandis que les surfaces implicites sont quant à elles surtout employées lors de la modélisation et visualisation médicale (Lorensen, W.E., Cline, H.E., 1997).
Dans le contexte du présente travail nous nous attarderons sur les représentations paramétrique et polygonale, qui sont les plus utilisées dans le domaine de l'ingénierie.

### 2.1.1 Surfaces paramétriques

La représentation paramétrique des surfaces a connu son développement dans les années 60-70 grâce à la croissance fulgurante des industries de l'automobile et de l'aéronautique. Cette représentation est basée sur un ensemble de relations (*une par coordonnée*) qui donne les valeurs des coordonnées dans l'espace Euclidien par une fonction de deux paramètres indépendants :

$$\vec{P}(u,v)=(x,y,z)$$
$$x = f(u,v)$$
$$y = g(u,v)$$
$$z = h(u,v)$$

Les paramètres u et v varient entre 0 et 1. (Camarero, 2003)

Du point de vue de l'approche paramétrique on peut représenter la surface par un assemblage de portions de surfaces (carreaux) imposant des conditions de raccord au niveau de la continuité entre les carreaux, i.e. que les carreaux ne doivent pas se superposer et qu'il ne peut y avoir de trous entre des carreaux connexes(Camarero, 2003).

#### 2.1.1.1 Représentation polynomiale

Une classe de fonctions particulièrement pratique pour la représentation paramétrique est la classe des polynômes qui présente un nombre important d'avantages en termes de facilité, de manipulation et de coût de calcul pour les évaluations. Dans la notation vectorielle, un point sur la surface peut être représenté par :

$$\vec{P}(u,\ v) = \sum_{i=0}^{p} \sum_{j=0}^{q} \vec{a}_{ij} u^i v^j$$ (notation inspirée de Guibault, 2005)

où : p représente le degré du polynôme / surface dans la direction u
q représente le degré du polynôme / surface dans la direction v

$\vec{a}_{ij}$ représente le vecteur des coefficients ($a_{ij}x$, $a_{ij}y$, $a_{ij}z$)

### 2.1.1.2 Surfaces NURBS et B-Splinaires

Les surfaces NURBS sont obtenues par produit tensoriel. Une surface obtenue par produit tensoriel est conceptuellement construite comme la double variation dans les directions u et v des quantités géométriques qui contrôlent la forme de la surface :

$$S(u,v)=F[u][G]H^{T}[v] \text{ (Guibault, 2005)}$$

où : F[u] représente le vecteur des fonctions de base en direction u

G est la matrice de données géométriques (*points de contrôle*)

H [v] est le vecteur transposé des fonctions de base en direction v

Les fonctions de base B-Splinaire normalisées de degré p sont définies récursivement par :

$$N_{i,0}(u)=\begin{cases} 1 \text{ si } u_i \le u < u_{i+1} \\ 0 \text{ sinon} \end{cases}$$

et

$$N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u) \text{ (Guibault, 2005)}$$

où : $u_i$ sont les nœuds du vecteur nodal U

Dans le contexte du produit tensoriel et des fonctions de base dans les directions u et v, une surface NURBS est définie par :

$$S(u,v)=\sum_{i=0}^{n_i}\sum_{j=0}^{n_j} N_{i,p}(u)\, N_{j,q}(v)\, P_{ij} \text{ (Guibault, 2005)}$$

où : $P_{ij}$ est une grille de points de contrôle pondérés par les vecteurs nodaux u et v

### 2.1.2 Représentation polygonale

La représentation polygonale des surfaces a connu un grand succès dans les années 70-80 avec le développement de la modélisation géométrique et les algorithmes de maillage dans le cadre de nouvelles disciplines comme la simulation numérique.

On peut définir une surface polygonale 3D comme un ensemble de polygones, le plus souvent un ensemble de triangles (Garland, 1999).

Ainsi, une surface polygonale est une partition physique qui possède des propriétés géométriques (*coordonnées des sommets*) et informations topologiques (*la connectivité*).

Finalement, nous pouvons résumer la définition d'un modèle polygonale : M=(V, F) qui est une combinaison de deux ensembles V et F où :

$V = (v_1, v_2, \ldots v_n)$ est un ensemble de sommets où chaque sommet peut être identifié par un entier unique i, où chaque sommet $v_i$ est représenté par un vecteur $v_i = [ x_i, y_i, z_i]$ dans l'espace Euclidien $R^3$ et;

$F = \{f_1, f_2, \ldots f_n\}$ est un ensemble ordonné de triangles où chaque triangle peut être identifié par un entier unique m, où chaque triangle $f_m = (j, k, l)$ est un triplet ordonné des indices identifiant les sommets $(v_j, v_k, v_l)$ du triangle $f_m$. (Garland, M. et al., 1997)

### 2.1.3 Caractère complémentaire des surfaces paramétriques et polygonales

Les surfaces paramétriques ont trouvé un grand champ d'application dans la modélisation géométrique pour la simulation numérique. Les avantages des surfaces paramétriques sont nombreux :

1. Exactitude de la description géométrique du modèle
2. Simplicité d'utilisation et notation compacte
3. Facilité d'obtention de surfaces lisses d'une forme complexe désirée avec des polynômes de degré 3 et plus
4. Souplesse de contrôle de la forme d'une surface à l'aide de ses points de contrôle

Même avec tous ces avantages, il n'est cependant pas possible de concevoir un modèle géométrique de forme complexe sans avoir recours aux notions d'entités topologiques comme une face, une arête ou une boucle, qui sont utilisées à titre d'éléments complémentaires afin de décrire la topologie d'un modèle. C'est pour cette raison que le logiciel Topovisu est basé sur les principes de la modélisation par les frontières du modèle (B-rep). La modélisation par les frontières, ou le modèle B-rep (Boundary Representation), est fondée sur le fait qu'un solide divise l'espace $R^3$ en deux ensembles continus de points séparés par la frontière (l'intérieur et l'extérieur). La représentation B-rep contient ainsi deux types d'informations : géométrique et topologique. L'information géométrique fournit la description paramétrique des courbes et des surfaces; l'information topologique quant à elle fournit les relations entre les sommets, les arêtes, les faces et les volumes. Dans ce contexte, un modèle B-rep est un ensemble ordonné de faces. Chaque face s'appuie sur une surface paramétrique, i.e. qu'une face est une partie de la surface délimitée par ses arêtes (décrites par les courbes paramétriques) qui forment une boucle de rognage.

Cette dernière détermine la forme d'une face. Voilà pourquoi il est aussi important, sinon plus, d'utiliser les surfaces paramétriques découpées comme modèles afin de tester les algorithmes de simplification. Ces modèles décrits paramétriquement seront transformés sous la forme polygonale par une opération de génération de maillage.

La raison pour laquelle nous nous concentrons sur la simplification des surfaces polygonales est purement pragmatique ; en effet les rendus graphiques produits par les librairies, telles OpenGL, sont faits sous forme polygonale.

De plus, il faut ici mentionner que lors de la simulation numérique, les modèles géométriques décrits paramétriquement sont convertis sous forme polygonale. La forme polygonale devient donc un complément important des surfaces paramétriques.

### 2.1.4 Surfaces variétés et non-variétés

Une caractéristique importante de la topologie d'un modèle est liée au fait que le modèle soit constitué de surfaces qui sont des variétés ou non. La plupart des objets qui font partie de notre quotidien et qui nous entourent peuvent être représentés par des surfaces variétés. Par exemple, le corps humain, le corps d'un animal ou la majorité des objets qui nous entourent sont des volumes fermés pouvant être décrits par des surfaces qui sont des variétés. Cependant, le feuillage des plantes, à titre d'exemple, peut être représenté à l'aide de surfaces qui ne sont pas des variétés

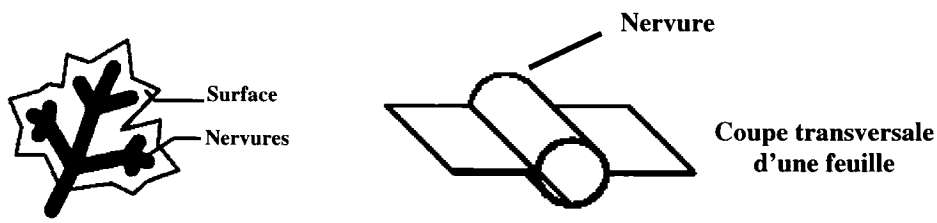
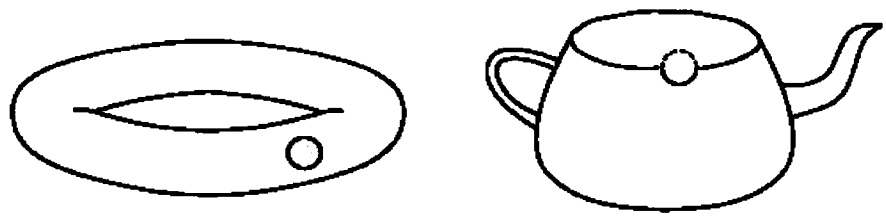



FIGURE 2.1 – Illustration de surfaces non variétés

Plus spécifiquement, une surface polygonale est une variété si ses sommets ont un voisinage équivalant à un disque, tandis qu'une surface est une variété avec frontières si

ses sommets internes ont un voisinage équivalant à un disque et à un demi disque sur les frontières(Cignoni, P.et al., 1997).

Une surface est dite une variété (avec frontières) si chaque arête de celle-ci a exactement deux triangles adjacents (à l'exception des arêtes sur la frontière qui devront n'être composées que d'un triangle adjacent) (Cignoni, P. et al., 1997).



**(a)** ***Variété*** **(b)** ***Variété avec frontières***

FIGURE 2.2 – Deux types de surfaces variétés

Plusieurs algorithmes de simplification procèdent par la répétition de simplifications locales du modèle alors que ces transformations locales pourront facilement résulter en une transformation d'une surface variété en une non-variété.

Il est donc important de mentionner dans le cadre du présent travail qu'il faut préserver la topologie initiale de la surface; nous ne devons donc pas permettre la transformation d'une surface variété en une surface non-variétés et vice et versa.

## 2.2 Survol des méthodes de simplification polygonale

Les deux méthodologies les plus communes dans la transformation des surfaces polygonales sont le **raffinement** et la **décimation**.

Un algorithme de raffinement prend en entrée une approximation grossière ou approximation à grande échelle et ajoute de nouveaux éléments à chaque itération.

Les algorithmes de décimation, ou de simplification, font exactement l'opération inverse; i.e. qu'ils commencent avec une surface originale et suppriment des éléments à chaque itération.

La plupart des méthodes itératives de simplification peuvent être classées en trois catégories, dépendamment de l'opérateur topologique utilisé. Ces trois catégories sont : décimation des polygones, décimation des sommets et contraction des arêtes.

### 2.2.1 Décimation des polygones

Les méthodes de décimation de polygones procèdent par la localisation d'un polygone et le remplissage de son espace par les polygones adjacents (Gieng, T.S. et al., 1997). Les résultats de ces méthodes sont généralement de piètre qualité (Hussain, M. et al., 2003).

### 2.2.2 Décimation des sommets

Une des méthodes les plus populaires de simplification de surfaces est la méthode de décimation des sommets, originalement proposée par Schroeder et ses associés en 1992.

À chaque itération un sommet est sélectionné et supprimé; le trou résultant est re-triangulé par la suite.

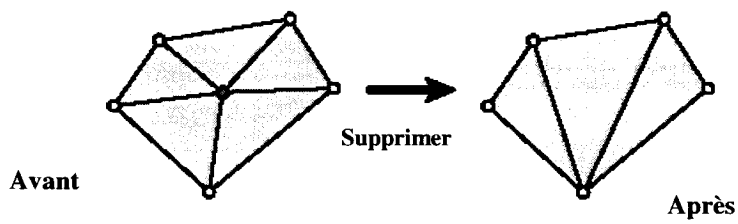



FIGURE 2.3 – Représentation schématique de l'opération de décimation d'un sommet

Puisque la re-triangulation exige la projection locale de la surface sur un plan, les algorithmes de décimation des sommets sont généralement limités aux surfaces variétés. Ainsi, les algorithmes de décimation des sommets sont capables de préserver la topologie initiale du modèle et de produire des approximations de bonne qualité.
Cependant, à cause du temps requis par le processus de re-triangulation, ces algorithmes sont lents (Hussain,M. et al., 2003).

### 2.2.3 Contraction des arêtes

La méthode de contraction des arêtes est un cas spécial de décimation des sommets mais ne requiert pas l'utilisation d'un processus de re-triangulation. C'est pour cette raison que les algorithmes de contraction des arêtes sont plus rapides et qu'il préservent de plus une bonne approximation du modèle initial (Hoppe, H., 1996).
Une opération de contraction des arêtes fusionne deux sommets d'une arête en un seul sommet; de ce fait elle soustrait deux triangles.

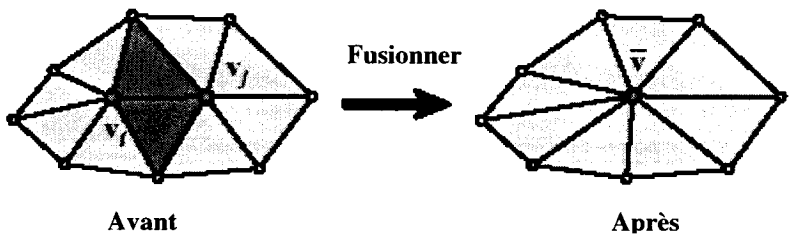



FIGURE 2.3 – Représentation schématique de l'opération de contraction

*L'arête ($v_i$,$v_j$) est contractée et remplacée par le point $v_j$, les triangles adjacents à l'arête ($v_i$,$v_j$) deviennent des triangles dégénérés qui sont éliminés.*

Pour simplifier une surface, l'opération de contraction des arêtes est appliquée de façon itérative et vorace jusqu'à ce que le nombre de triangles désiré ou la tolérance d'erreur désirée soit obtenu.

Pour ce faire, nous devons prendre deux décisions; la première sera de prioriser les contractions qui mèneront à une « bonne » solution sans que celle-ci soit nécessairement optimale et la deuxième décision sera de choisir les sommets qui remplaceront ceux qui auront été supprimés (Garland, M., 1999).

Quant au choix des sommets de remplacement, deux approches sont possibles; la première consiste à placer les sommets résultants à la position d'un des deux sommets supprimés.

Il s'agit ici de la stratégie la plus simple, laquelle est communément appelée « *half-edge collapse* ».

La deuxième possibilité, quant à elle, consiste à trouver la position localement optimale des sommets résultants; i.e. que les sommets résultants ne sont pas nécessairement placés à la position de l'un des deux sommets supprimés mais pourront être placés librement dans l'espace, ce qui a pour but de minimiser la déformation par rapport à la surface originale.

En ce qui concerne la priorisation des contractions désirées, elle est déterminée de façon automatique par une mesure d'erreur qui reflète la déviation géométrique locale. La façon dont cette erreur est calculée différencie les algorithmes les uns des autres. La politique de mesure d'erreurs peut être locale ou globale.

Il est important de noter que les algorithmes de fusion des sommets peuvent implicitement altérer la topologie d'une surface polygonale par la fermeture de trous dans la surface. Pour éviter cela, différentes techniques de préservation de topologie peuvent être appliquées.

## 2.3 Évaluation de la qualité de l'approximation des surfaces

Tel que mentionné précédemment, le but premier de la simplification est de produire l'approximation la plus similaire possible à une surface originale, tout en réduisant le nombre total de sommets. Dans le but de s'assurer de la qualité d'une approximation, nous devrons recourir aux notions quantitatives de similarité.
Supposons que nous avons deux modèles polygonaux, $M_1$ et son approximation $M_2$ ; nous désirons obtenir la métrique d'erreur E, soit :

$$E : |M_1 - M_2| \rightarrow \Re$$

pour laquelle la valeur E ($M_1$, $M_2$) mesure l'erreur d'approximation de $M_2$.
La similarité de $M_2$ à $M_1$ est alors inversement proportionnelle à la valeur d'erreur assignée par E à $M_2$.

Plusieurs métriques d'erreur d'approximation existent de nos jours; entre autres les trois suivantes :

1. Déviation des formes
2. Déviation des attributs (normales, couleurs)
3. Déviation des textures

En ce qui concerne le présent travail, le choix du premier critère de la qualité d'approximation s'arrête naturellement sur la similarité de forme, que l'on peut également nommer similarité d'apparence.

### 2.3.1 Similarité d'apparence

Le domaine d'intérêt principal du présent travail est étroitement relié à la visualisation et au rendu graphique. C'est pour cette raison que la similarité d'apparence est devenue le critère principal pour évaluer la qualité d'approximation.

Étant donné que la plupart des algorithmes de simplification sont basés sur un critère géométrique, nous présenterons la notion d'erreur d'approximation géométrique.

### 2.3.1.1 Erreur d'approximation géométrique

Puisque la similitude d'apparence est un objectif primordial lors de la simplification des surfaces, il est plus facile de considérer la similitude géométrique comme le remplacement valide de la similitude d'apparence.

Avant de nous concentrer sur les problèmes de mesures d'approximation sur les modèles polygonaux, nous désirons débuter par des cas plus simples, comme l'approximation des fonctions.

Les deux métriques d'erreurs les plus communes sont les normes $L_\infty$ et $L_2$. (Garland, M. et al., 1997)

Supposons une fonction de valeurs réelles f(t) et une approximation g(t), et un intervalle d'intérêt [a,b] ; alors la norme $L_\infty$ qui mesure la déviation maximale entre la fonction originale et son approximation est définie par :

$$\|f-g\|_\infty = \max_{a \le t \le b} |f(t)-g(t)|$$

La norme $L_2$ où l'erreur quadratique moyenne est définie par :

$$\|f-g\|_2 = \sqrt{\int_a^b (f(t)-g(t))^2 dt}$$

laquelle fournit la mesure de déviation moyenne entre les deux fonctions.

L'approximation linéaire par morceaux g(t), composée de *n* segments, est appelée optimale s'il n'existe aucune autre approximation de *n* segments ayant une erreur inférieure.

### 2.3.2 Distance de Hausdorff

Nous pouvons formuler les représentations analogues de $L_\infty$ et $L_2$ pour les surfaces.
Lors de la comparaison des surfaces, la distance est mesurée entre les paires de points les plus proches. Si nous définissons par P(M) l'ensemble de tous les points sur la surface M, alors la distance du point *v* à la surface M est définie par la distance au point le plus proche sur la surface M :

$$d_v(M)=\min_{w\in P(M)}\|v-w\|$$ (Aspert, N. et al., 2002)

La distance de Hausdorff est la métrique de mesure d'erreur géométrique la plus commune; sa correspondance la plus proche est la métrique $L_\infty$.
Alors la distance de Hausdorff $E(M_1, M_2)$ entre la surface originale $M_1$ et l'approximation $M_2$ est définie par :

$$E(M_1,M_2)=\max_{v\in P(M_1)} d_v(M_2)$$ (Aspert, N. et al., 2002)

Il est à noter que la distance de Hausdorff est non symétrique, i.e que $E(M_1, M_2)$ n'est pas égal à $E(M_2, M_1)$. La distance $E(M_1, M_2)$ est appelée la distance « avant » et la distance $E(M_2, M_1)$ est appelée « arrière ».(Aspert, N. et al., 2002)

Il est plus convivial d'introduire la distance de Hausdorff symétrique :

$$E_{max}(M_1,M_2)=\max\left[\max_{v\in P(M_1)} d_v(M_2),\max_{v\in P(M_2)} d_v(M_1)\right]$$ (Aspert, N. et al., 2002)

En effet, celle-ci fournit les mesures maximales de déviation entre les deux surfaces; ainsi la mesure est plus exacte comparativement à la distance non symétrique, tel qu'illustré par la figure ci-dessous :

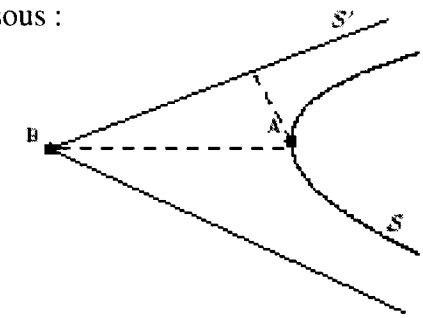



FIGURE 2.4 – Illustration de la distance d'Hausdorff entre les deux surfaces

Dans le présent cas, la mesure E(S, S') demeurera inférieure à la mesure E(S', S) puisque que $d_A(S') << d_B(S)$ (Aspert, N. et al., 2002)

### 2.3.3 Erreur moyenne et erreur quadratique moyenne entre deux surfaces

En utilisant la définition de la distance point-surface, nous pouvons exprimer l'erreur quadratique moyenne entre deux surfaces :

$$E_{qmoy}(M_1,M_2)=\sqrt{\frac{1}{w_1+w_2}\left(\int_{v\in P(M_1)} d_v^2(M_2)dv+\int_{v\in P(M_2)} d_v^2(M_1)dv\right)}$$

où $w_1$ et $w_2$ représentent les superficies des deux surfaces respectives,
et l'erreur moyenne :

$$E_{moy}(M_1,M_2)=\frac{1}{w_1+w_2}\left(\int_{v\in P(M_1)} d_v(M_2)dv+\int_{v\in P(M_2)} d_v(M_1)dv\right)$$

Dans le cas des surfaces polygonales, le logiciel MESH simplifie les calculs des intégrales par l'échantillonnage de *dv* sur un ensemble de points discrets.

Étant donné que nous avons $P(M_1)$ et $P(M_2)$ représentant les ensembles de tous les points situés sur les surfaces $M_1$ et $M_2$, nous pouvons alors choisir les deux sous-ensembles $X_1$ et $X_2$ avec le nombre de points $k_1$ et $k_2$ respectivement. Ces ensembles doivent contenir comme minimum tous les sommets de leur modèle respectif. Les métriques d'erreurs représentées ci-haut sous la forme discrète et échantillonnée sont :

$$E_{max}(M_1,M_2)=\max\left[\max_{v\in X_1} d_v(M_2),\max_{v\in X_2} d_v(M_1)\right],$$

$$E_{qmoy}(M_1,M_2)=\sqrt{\frac{1}{k_1+k_2}\left(\sum_{v\in X_1} d_v^2(M_2)+\sum_{v\in X_2} d_v^2(M_1)\right)},$$

et

$$E_{moy}(M_1,M_2)=\frac{1}{k_1+k_2}\left(\sum_{v\in X_1} d_v(M_2)+\sum_{v\in X_2} d_v(M_1)\right)$$

On peut remarquer la construction symétrique de toutes les métriques présentées. Le logiciel MESH offre la possibilité de mesurer l'erreur maximale, l'erreur minimale, l'erreur quadratique moyenne et l'erreur moyenne sous la forme symétrique, ce qui augmente la précision des mesures.

# CHAPITRE 3

# SURVOL DES TRAVAUX PRÉCÉDENTS

Dans ce chapitre, nous examinerons quelques travaux publiés en ce qui concerne la comparaison d'algorithmes de simplification et justifierons la pertinence des algorithmes choisis et des cas tests choisis. Nous verrons également à justifier la pertinence du choix du logiciel MESH comme outil d'évaluation numérique de la qualité des surfaces simplifiées.

## 3.1 Travaux sur la comparaison des méthodes de simplification

Malgré l'abondance de la littérature sur la simplification des surfaces, nous ne sommes toutefois pas en mesure de citer un grand nombre de travaux qui concerne la comparaison des algorithmes de simplification. La plupart des travaux disponibles dans le domaine public à ce sujet sont surtout des articles décrivant des méthodes existantes où nous trouverons des descriptions de méthodologies de simplification sans comparaison exhaustive entre les méthodes. Pour les revues de méthodes de simplification, se référer au travail de M. Garland (*Quadric-Based Polygonal Surface Simplification,1999*), section 2.4.

En effet, nous n'avons trouvé que quelques travaux qui contiennent des comparaisons quantitatives d'algorithmes. (Cignoni, P. et al., 1997 / Lindstrom, P. et al., 1999 / Garland, M. et al., 1997 / Hussein, M. et al., 2003)

P. Cignoni et ses collaborateurs furent en 1997 parmi les premiers à comparer la complexité empirique des algorithmes et la qualité d'approximation des surfaces résultantes.

Dans leur travail ils comparent six algorithmes de simplification de surfaces, soit « Mesh Decimation », « Simplification Envelopes », « Multiresolution Decimation », « Mesh Optimization », « Progressive Meshes » et « Quadric Error Metrics Simplification ».

Les six implémentations mentionnées ci-haut ont été testées sur trois modèles de la manière suivante: la simplification est effectuée selon sept niveaux de détails; par exemple, selon le taux de réduction de triangles à 15 %, 30 %, 50 %, 60 %, 70 % et 80 %. Le temps requis pour la simplification est également pris en compte. Par la suite, les erreurs $E_{max}$ et $E_{moy}$ ont été évaluées sur les modèles simplifiés.

Les meilleurs résultats en terme d'erreur $E_{moy}$ ont été obtenus par les algorithmes « Progressive Meshes » et « Mesh Optimization » et ce, dû au fait que ces deux algorithmes sont basés sur les métriques d'évaluation de $L_2$. En ce qui concerne l'erreur $E_{max}$, les algorithmes « Simplification Envelopes » et « Multiresolution Decimation » ont produit les meilleurs résultats.

P. Cignoni et ses collaborateurs notent la rapidité et la précision de l'algorithme « Quadric Error Metrics Simplification » de M. Garland.

Finalement, ils concluent que les algorithmes basés sur la métrique d'évaluation $L_\infty$ produisent les meilleurs résultats quant à l'évaluation de l'erreur $E_{max}$ et se comparent favorablement aux algorithmes basés sur la métrique d'évaluation $L_2$, quant à l'évaluation de l'erreur $E_{moy}$.

On peut reprocher à P. Cignoni et ses collaborateurs de n'avoir utilisé que trois modèles dans leur travail de comparaison; le nombre de modèles étant réduit, les résultats ne sont pas très significatifs.

Dans le travail de G. Turk et P. Lindstrom (1999), ces mêmes algorithmes ont été choisis, à l'exception de l'algorithme « Mesh Decimation » qui a été remplacé par l'algorithme « Memoryless Simplification ». La comparaison numérique a été effectuée sur deux modèles; un simple (sphère) et un complexe (cheval). Le modèle de la sphère composé de 20,480 faces a été créé par le procédé de subdivision à partir d'un icosaèdre comportant vingt faces. Étant donné que les auteurs avaient initialement identifié le but à atteindre, soit un modèle simplifié comportant vingt faces, un tel modèle simplifié a été créé à partir de chaque algorithme.

Par la suite les auteurs ont identifié les algorithmes produisant une simplification plus près par sa forme du modèle initial, soit de l'icosaèdre. Pour ce critère, l'algorithme le plus performant fut identifié comme étant « Mesh Optimization » lequel a produit un icosaèdre presque parfait.

À notre avis ce type de test ne reflète que très partiellement les performances d'un algorithme de simplification car un algorithme de simplification basé sur la subdivision inversée peut produire des résultats parfaits pour ce genre de test. La méthodologie de comparaison reste toutefois intéressante bien que les résultats ne puissent pas s'appliquer directement au type de surface qui nous intéresse dans le contexte du présent travail.

Le deuxième critère fut d'identifier comment l'algorithme « Memoryless Simplification » préserve les volumes initiaux du modèle, dépendamment des algorithmes de positionnement des sommets. Voici la liste complète des algorithmes de positionnement de sommets que l'algorithme « Memoryless Simplification » peut utiliser :

1. Aléatoire
2. Optimale
3. Centre de l'arrête
4. Optimisation du volume

5. Préservation du volume avec placement aléatoire
6. Préservation du volume avec placement optimal
7. Préservation du volume avec placement au centre de l'arrête
8. Préservation du volume avec optimisation du volume

Pour ce genre de test, les auteurs en sont arrivés à la conclusion que les méthodes utilisant les algorithmes 3 et 7 sont moins performantes que les autres méthodes énumérées. Les algorithmes de positionnement 2 et 6 produisent de meilleures performances que les méthodes 1 et 5.

En conclusion, les méthodes utilisant la contrainte de préservation du volume, soit les méthodes, 5, 6, 7 et 8, offrent de meilleures performances que les méthodes 1, 2, 3 et 4.

Le troisième critère fut d'évaluer comment les algorithmes préservent la forme des boucles/frontières. Pour ce faire, les auteurs ont simplifié deux modèles contenant des boucles/frontières, leur critère étant d'évaluer de quelle façon les trous dans la surface étaient préservés. Les résultats obtenus pour ce genre de tests furent les suivants :

Deux des méthodes, soit « Multiresolution Decimation » et « Quadric Error Metrics Simplification », transfèrent le plus grand nombre de polygones vers les boucles/frontières, produisant ainsi de meilleurs résultats.

Les méthodes « Progressive Meshes » et « Simplification Envelopes » utilisent quant à elles un plus petit nombre de polygones autour des boucles/frontières, produisant donc ainsi des résultats moins satisfaisants.

Finalement, la méthode « Mesh Optimization » produit les résultats les moins performants.

Le quatrième critère fut le temps d'exécution. Pour procéder à ce test, un modèle représentant un cheval a été simplifié selon huit niveaux de simplification et le temps consommé par chacun des algorithmes a été évalué. Les résultats pour le temps d'exécution sont les suivants :

La première place en terme de meilleurs résultats est occupée par « Quadric Error Metrics Simplification », suivi par « Memoryless Simplification », « Multiresolution Decimation », « Simplification Envelopes » et « Mesh Optimization ». De plus, les auteurs affirment que les algorithmes « Simplification Envelopes », « Mesh Optimization » et « Memoryless Simplification » ont un temps d'exécution presque constant indépendamment du degré de simplification.

Le cinquième critère fut la comparaison selon la distance de Hausdorff. Les meilleurs résultats en terme d'erreur $E_{moy}$ ont été obtenus par les algorithmes « Mesh Optimization » et « Memoryless Simplification ». Quant à l'erreur $E_{max}$, les algorithmes « Simplification Envelopes » et « Multiresolution Decimation » ont produit les meilleurs résultats. Comme on peut le constater, les résultats de cet article concordent avec ceux de l'article précédent.

En conclusion, les auteurs affirment que leur algorithme « Memoryless Simplification » performe mieux lors de la comparaison selon la métrique d'erreur $E_{moy}$ mais laisse à désirer lorsqu'il s'agit de l'erreur $E_{max}$. En plus, la consommation de temps de cet algorithme ne dépend pas forcément du degré de simplification du modèle et reste élevée. Pour ces raisons nous n'aurons pas recours à cet algorithme dans notre travail.

Dans le troisième article, M. Hussain et ses collaborateurs (Hussain, M. et al., 2003), comparent leur algorithme aux algorithmes suivants : « Quadric Error Metrics Simplification », « Memoryless Simplification », « Multiresolution Decimation » et « Simplification Envelopes » et ont effectué des tests sur deux modèles complexes.

Les comparaisons des algorithmes furent effectuées selon les critères suivants : le temps d'exécution, l'erreur $E_{moy}$ et l'erreur $E_{max}$ .

Quand au temps d'exécution, les auteurs affirment que leur algorithme est deux fois plus lent que l'algorithme « Quadric Error Metrics Simplification ». Les autres algorithmes furent encore plus lents. Par exemple, l'algorithme « Memoryless Simplification » fut cinq fois plus lent, alors que l'algorithme « Multiresolution Decimation » fut dix fois plus lent et finalement, l'algorithme « Simplification Envelopes » fut dix-sept fois plus lent que l'algorithme « Quadric Error Metrics Simplification ».

En ce qui concerne la comparaison selon la métrique d'erreur $E_{moy}$, leur algorithme performe mieux que tous les autres, à l'exception cependant de l'algorithme « Memoryless Simplification ».

Quant à la comparaison selon la métrique d'erreur $E_{max}$, leur algorithme se compare favorablement avec les autres, à l'exception toutefois de l'algorithme « Multiresolution Decimation ».

Ils en viennent donc à la conclusion que parmi les algorithmes existants, le leur se classe deuxième en terme de rapidité, soit immédiatement après l'algorithme « Quadric Error Metrics Simplification ». Cependant, en ce qui concerne la consommation de mémoire, leur algorithme consomme au moins quarante octets de moins par sommet que l'algorithme « Quadric Error Metrics Simplification ». Quant à l'exactitude numérique de l'algorithme, elle demeure très bonne.

Tel que mentionné précédemment, toutes les recherches de comparaison des algorithmes précitées furent limitées à quelques modèles seulement. Les résultats ainsi obtenus ne sont pas vraiment représentatifs alors qu'un grand nombre de modèles aurait été souhaitable, produisant ainsi des résultats plus fiables.

Pour cette raison nous avons opté pour une méthodologie de comparaison plus exhaustive, soit en travaillant sur un grand nombre de surfaces, ce qui selon nous a produit des résultats scientifiquement plus significatifs.

## 3.2 Justification du choix du logiciel MESH

Tel que mentionné à la section précédente, P. Cignoni et ses collaborateurs ont utilisé le logiciel Metro afin d'effectuer la comparaison des surfaces (Cignoni, P. et al., 1998). Ce logiciel est utilisé principalement pour l'évaluation d'erreurs introduites lors de simplification de surfaces. À l'aide de cet outil, il est possible d'évaluer l'amplitude et la distribution spatiale de l'erreur numérique. Les valeurs de l'erreur numérique sont calculées selon la définition de la distance d'Hausdorf dont il est question dans la section 2.3.2 du présent travail. Les calculs d'erreurs sont faits en deux étapes :

1. Échantillonnage dans un ensemble de points discrets de tous les triangles de la surface non simplifiée
2. Calcul de la distance d'Hausdorf des points échantillonnés de la surface simplifiée à la surface non simplifiée

Le logiciel MESH (Aspert, N. et al., 2002) quant à lui possède toutes les fonctionnalités du logiciel Metro mais donne en plus la possibilité d'évaluation d'une métrique d'erreurs supplémentaire $E_{qmoy}$, aussi représentée à la section 2.3.2. Le logiciel MESH offre de plus une rapidité d'évaluation d'erreurs accrue comparativement au logiciel Metro (Aspert, N. et al., 2002).

### 3.3 Justification des algorithmes choisis

Dans tous les travaux en référence (selon section 3.1) malgré le grand nombre d'algorithmes comparés et leur diversité, un algorithme est commun à tous ces travaux, soit l'algorithme « Simplification des surfaces via l'erreur quadratique » de M. Garland et tous les auteurs l'ont unanimement évalué comme étant le plus rapide et son rapport coût/précision a été évalué comme très avantageux comparativement aux autres algorithmes existant à ce jour.

C'est donc pour cette raison que notre choix premier d'algorithme à implémenter s'est arrêté sur la version la plus récente et améliorée par la pondération de la superficie des triangles (Garland, M., 1999)

Quant au choix du deuxième algorithme à implémenter, notre choix s'est arrêté sur l'algorithme le plus récent, soit « Simplification des surfaces via l'erreur géométrique » (Hussain, M. et al., 2004).

Ce choix fut justifié par:

1. L'exactitude numérique
2. La préservation de la topologie initiale des surfaces
3. La consommation de mémoire réduite

...le tout, comparativement à l'algorithme précédent de M. Garland

En effet, la consommation de mémoire lors de l'intégration d'algorithmes dans un logiciel de grande taille jouera un rôle primordial dans la performance et la fonctionnalité dudit logiciel surtout s'il est utilisé sur un ordinateur à performance moyenne ou réduite.

Quant au choix du troisième algorithme, soit « Simplification des surfaces via l'erreur de courbure discrète » (Kim, S.-J. et al., 2002), nous n'avons pas trouvé dans la littérature de comparaison de cet algorithme avec les autres étant donné qu'il est relativement nouveau.

Cependant, nous l'estimons comme un algorithme prometteur, basé sur une métrique d'erreur complètement différente de celles des deux algorithmes précédents. Pour cette raison nous avons tenu à comparer sa précision en terme géométrique aux deux autres algorithmes. De plus, nous comptons l'améliorer en remplaçant l'opérateur d'évaluation de courbure moyenne (Meyer, M. et al., 2002) alors que dans le cas de sommets frontaliers, nous ajouterons une métrique d'erreurs basée sur la distance géométrique (algorithme de M. Hussain).

### 3.4 Justification des cas tests choisis

Dans tous les travaux auxquels il est fait référence ci-haut, la comparaison numérique est effectuée selon deux métriques d'erreurs : l'erreur maximale $E_{max}$ et l'erreur moyenne $E_{moy}$. La méthodologie de comparaison numérique est commune aux quatre travaux ; l'atout principal de cette méthodologie est la présence de seulement 2 ou 3 modèles sur lesquels la simplification est effectuée selon 7 ou 8 niveaux de détails *(par exemple selon le taux de réduction de triangles à 15 %, 30 %, 50 %, etc.)*. Le temps requis pour la simplification est également pris en compte. Par la suite, les normes d'erreurs sont évaluées pour chaque niveau de détails.

Dans le cadre du présent travail, nous proposerons une autre méthodologie d'évaluation numérique.

1. Nous ferons des **évaluations sur une centaine de modèles** de surfaces rognées et obtenues à l'aide de l'interface de rappel d'OpenGL, ce qui rendra nos comparaisons des algorithmes statistiquement plus fiables comparativement aux travaux effectués précédemment. La quantité de surfaces utilisées a été déterminée par la disponibilité de celles-ci à notre disposition.

2. En ce qui concerne le temps d'exécution, nos résultats de comparaison utiliseront une implémentation de tous les algorithmes sur une plate-forme commune, ce qui n'a pas été fait dans les travaux discutés précédemment.
   L'implémentation sur une même base est plus valide étant donné l'assurance que nous n'utiliserons que les mêmes ressources disponibles pour chacun des algorithmes ; nous serons donc assurés de ne pas voir de présences ou d'absences d'opérations coûteuses, par exemple lecture ou écriture sur disque dur dans un ou plusieurs algorithmes, comme cela pourrait survenir dans le cas où les implémentations distinctes des algorithmes sont testées.

3. Nous ajouterons aux deux métriques d'erreurs figurant dans les travaux précédents une troisième métrique, soit l'erreur quadratique moyenne $E_{qmoy}$.

4. La distance d'Hausdorff symétrique sera utilisée pour évaluer les trois types d'erreurs.

# CHAPITRE 4

# DESCRIPTION DES TROIS ALGORITHMES DE SIMPLIFICATION

Le présent chapitre décrira en détail les algorithmes choisis pour cette étude, tel que mentionné au chapitre précédent.

Le but de la simplification des surfaces est de prendre à titre de donnée un modèle polygonal et d'en générer un modèle simplifié i.e. en faire une copie approximative. Nous présumons que les modèles d'entrée sont présentés sous forme de maillages triangulaires. Les modèles d'approximation devront répondre à certains critères, soit par un nombre donné de triangles restants ou selon un certain seuil d'erreurs acceptable.

Tous les algorithmes utilisés dans le présent travail simplifient les surfaces par contraction itérative des arêtes. Une paire (v1, v2) de sommets est une paire candidate pour la contraction dès que le critère suivant est respecté :

- *(v1, v2) est une arête*

De plus, le choix de la contraction suivante sera effectué lors du déroulement de l'algorithme; nous devrons introduire la notion du *coût* de contraction, méthode qui permettra d'établir la différence entre tous les algorithmes utilisés.

Voici un algorithme générique de simplification de maillages :

1. Définir les candidats pour l'opération de fusion des arêtes;
2. Trier la liste des candidats selon les coûts de contraction;
3. Effectuer l'opération de fusion de sommets; les triangles dégénérés seront éliminés du maillage;
4. Mettre à jour la liste des candidats pour l'opération de fusion (recalcule des coûts);
5. Passer à l'étape 3 si des candidats pour l'opération de fusion demeurent.

## 4.1 1$^{er}$ algorithme – Simplification des surfaces via l'erreur quadratique

Le but de cet algorithme est de produire une approximation de haute qualité des surfaces tout en conservant la rapidité d'exécution. Avant le développement de cet algorithme aucun ne pouvait fournir un bon compromis entre la qualité d'approximation et la rapidité d'exécution des calculs (Garland, M., 1999).

L'utilisation de cet algorithme suppose que l'approximation n'a pas besoin de conserver la topologie de la surface originale. Pourtant, on pourrait facilement nommer ici plusieurs applications où la topologie de la surface fournit de l'information critique.

Ainsi, cet algorithme permet la simplification des surfaces *non-variétés* et permet de transformer les surfaces variétés en approximation *non-variétés* (Garland, M., 1999).

### 4.1.1 Contraction itérative des arêtes

Chaque itération de cet algorithme implique l'application d'une opération atomique: la contraction d'une arête. Cette contraction décrite par $(v_i, v_j) \rightarrow v$ modifie la surface en trois étapes :

1. Fusionne les sommets $v_i$ et $v_j$ à la position v
2. Remplace toutes les occurrences du sommet $v_j$ en sommet $v_i$
3. Enlève tous les triangles dégénérés

*À noter que la première étape modifie la géométrie de la surface tandis que les deuxième et troisième étapes modifient la connectivité de la surface* (Garland, M., 1999).

Comme la plupart des algorithmes concernés, le présent algorithme se catégorise parmi les algorithmes « voraces ». La séquence de contraction des arêtes est déterminée de façon vorace, i.e. une fois qu'une arête est sélectionnée pour la contraction, il s'agit d'une étape définitive pour l'arête puisqu'elle ne sera plus prise en considération à n'importe quelle étape ultérieure. Sommairement, l'algorithme est décrit par les étapes suivantes : (Garland, M., 1999).

1) Sélection d'un ensemble de candidats/sommets pour la contraction
2) Attribution des coûts de contraction à chaque candidat
3) Placement de tous les candidats dans une liste (*last in first out*) selon le coût minimal de contraction
4) Répétition jusqu'à ce que le niveau d'approximation désiré soit atteint :
   a) Élimination d'une arête $(v_i , v_j)$ selon le coût minimal de contraction
   b) Fusion de la paire $(v_i , v_j)$
   c) Mise à jour des coûts de tous les candidats voisinant $v_i$

### 4.1.2 Sélection de candidats

La sélection des candidats est effectuée en fonction des caractéristiques des arêtes i.e. que toutes les paires de sommets reliés par une arête sont sélectionnées, tel que mentionné précédemment dans ce chapitre.

### 4.1.3 Calcul et assignation des coûts de contraction

Les coûts de contraction de cet algorithme sont basés sur la métrique d'erreurs de la distance quadratique. Voici donc ci-dessous les explications relatives au calcul des coûts en question.

Dans le maillage original, chaque sommet est le résultat de l'intersection d'un ensemble de plans, à savoir les plans des triangles adjacents au sommet. Nous pouvons associer ledit ensemble de plans à chacun des sommets du maillage et également définir une erreur associée à ce sommet en se basant sur la déviation locale des plans par rapport à un plan moyen.
Cette erreur représente la somme des distances au carré d'un sommet donné, à l'ensemble de plans qui y sont associés.

$$\Delta(v) = \Delta([v_x\ v_y\ v_z]^T) = \sum_{p \in planes(v)} (p^T v)^2$$

où « $p^T = [abcd]^T$ » représente le plan défini par l'équation suivante :

$$ax + by + cz + d = 0$$

et

$$a^2 + b^2 + c^2 = 1$$

L'erreur quadratique peut être réécrite sous la forme :

$$\Delta(v) = \sum_{p \in planes(v)} (v^T p)(p^T v) = \sum_{p \in planes(v)} v^T (pp^T)\, v = v^T \left( \sum_{p \in planes(v)} \mathbf{K_p} \right) v$$

où $\mathbf{K_p}$ représente la matrice 4X4 :(Garland, M. et al., 1997)

$$\mathbf{K_p} = pp^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Cette matrice peut être utilisée pour calculer la distance au carré de n'importe quel point dans l'espace au plan **p**.

Nous pouvons également représenter par une matrice **Q** la somme des matrices $\mathbf{K_p}$ pour l'ensemble des plans associés à un sommet $\mathbf{v} = [v_x, v_y, v_z, 1]^T$ :

$$\mathbf{Q} = \sum_{i=0}^{n} \mathbf{K_{pi}}$$

où **i** est l'index du triangle voisinant.

Nous définissons alors les coûts (erreur) pour un sommet **v** comme :

$$\Delta(v) = v^T Qv$$

### 4.1.4 Choix de la position des sommets résultants

Il est à mentionner que l'algorithme permet dans le cas où la matrice **Q** est réversible, d'optimiser l'emplacement du sommet $\overline{\mathbf{v}}$ résultant de la contraction, i.e. trouver la position $\overline{\mathbf{v}}$ pour laquelle $\Delta(\overline{\mathbf{v}})$ sera minimal.

Pour trouver $\overline{\mathbf{v}}$ , nous devons résoudre :

$$\overline{\mathbf{v}} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Le choix du positionnement des sommets résultants se déroule en trois étapes :

1. Résolution du système décrit plus haut ;
2. Si la matrice **Q** est non inversible, trouver un emplacement optimal sur l'arête ($v_i$, $v_j$) ;
3. Si la position optimale n'est pas unique (**Q** est singulière), alors choisir l'emplacement sur un des sommets $v_i$ ou $v_j$ (Garland, M., 1999).

### 4.1.5 Simplification des surfaces via l'erreur quadratique avec pondération par la superficie des triangles

La méthode précédente peut être améliorée par la pondération de la matrice **Q** par un poids quelconque. Sur la figure 4.1, on représente la division d'un triangle en trois fragments associés à chacun des sommets

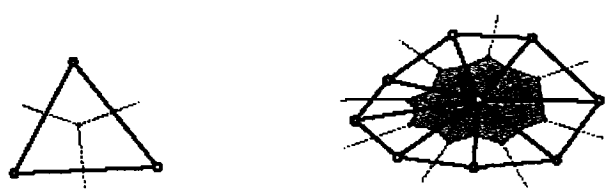



FIGURE 4.1 – Pondération de la matrice quadratique par les superficies des triangles

La pondération par la superficie est faite en deux étapes :

1. Chaque triangle divisé en trois fragments
2. Le sommet central cumule le poids de tous les triangles adjacents. Essentiellement la différence entre le présent algorithme et le précédent réside dans la façon de calculer la matrice $\mathbf{K_p}$ :

$$\mathbf{K_p} = \mathbf{pp}^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \mathbf{w_i}$$

où $\mathbf{w_i}$ représente le poids basé sur la superficie du triangle associé au plan **p.**

La représentation mathématique du terme $\mathbf{w_i}$ est par défaut dans l'implémentation :

$$\mathbf{w_i = w / 3}$$

mais on peut améliorer encore ce terme par la formule suivante :

$$\mathbf{w_i = w\,\theta_i / \pi}$$

où **w** est la superficie du triangle **i** et $\theta$ est l'angle du coin du sommet en question (Garland, M., 1999).

Le reste de l'algorithme demeure le même que pour l'algorithme précédent.

Cette amélioration permet de faire en sorte que la somme des matrices $\theta_i$ pour une région donnée demeure invariante face aux triangulations différentes de cette région.

### 4.1.6 Description de l'algorithme de M. Garland

En s'appuyant sur la description précédente des détails spécifiques de l'algorithme, nous présentons ici la description schématique détaillée par étapes :

**Étape 1 :**

1. Sélectionner toutes les paires de candidats ($v_i$, $v_j$), tel que ($v_i$, $v_j$) est une arête
2. Allouer une matrice $Q_i$ pour chacun des candidats $v_i$
3. Pour chaque triangle $F_i$ = (j, k, l), calculer la matrice $Q_i$ . Additionner cette matrice aux matrices respectives des sommets j, k et l, pondérées adéquatement (voir section précédente)
4. Pour chaque paire de candidats ($v_i$, $v_j$) :
   a) *Calculer la matrice $Q = Q_i + Q_j$*
   b) *Sélectionner la position du sommet résultant v' (voir section « Choix de la position des sommets résultants »*
   c) *Appliquer les pénalités aux sommets se trouvant sur les frontières*
   d) *Placer la paire de candidats dans une file basée sur le coût de Q(v')*

**Étape 2 :**

5. Répéter jusqu'à ce que le niveau d'approximation désiré soit atteint.
   a) *Enlever la paire ($v_i$, $v_j$) avec le coût minimal de la file*
   b) *Effectuer la contraction ($v_i$, $v_j$) → v'*
   c) *Affecter $Q_i = Q_i + Q_j$*
   d) *Pour chaque sommet voisin $v_i$ ($v_i$, $v_k$), calculer les positions et les coûts comme au point 4; mettre à jour la file.*

(Garland, M., 1999)

## 4.2 2e algorithme - Simplification des surfaces via l'erreur géométrique

Cet algorithme est plus efficace en ce qui concerne l'utilisation de la mémoire et nécessite un minimum d'opérations de calcul. Il est basé sur les mesures globales de déviation géométrique et utilise une valeur associée à l'importance visuelle de chaque sommet afin de prévenir la décimation des parties visuellement importantes, de même que les parties à niveau élevé de détails.

### 4.2.1 Calcul de l'importance visuelle d'un sommet

L'importance visuelle d'un sommet est définie par la formule

$$w_v = 1 - \| k_v \|$$

où « k » représente la somme des produits de la superficie et des vecteurs normaux des triangles voisins divisée par la somme de la superficie des triangles voisins du sommet :

$$k_v = \sum \Delta_i \, n_i / \sum \Delta_i$$

L'importance visuelle représente dans une certaine mesure la courbure locale de la surface. Si l'importance visuelle d'un sommet est égale à zéro, ce sommet est considéré comme appartenant à une région localement plane de la surface. Si l'importance visuelle est différente de zéro, il ne s'agit donc pas d'un sommet plat et on doit donc être prudent lors de la suppression de ce sommet.

### 4.2.2 Calcul de l'erreur géométrique :

Lors de la suppression d'une arête du maillage « $e(v_{from}, v_{to})$ », un triangle du maillage devient un triangle dégénéré. Quant au reste des triangles adjacents au sommet « $v_{from}$ », ils seront modifiés et échangeront leur « $v_{from}$ » pour « $v_{to}$ » *(représentés sur la figure 4.2 par $v_0$ et $v_i$ respectivement).*

Lors de cet échange, étant donné que le positionnement géométrique de « $v_{to}$ » est différent de « $v_{from}$ », ces triangles subiront un changement de superficie et une rotation autour de l'axe des deux sommets restants et non impliqués dans la suppression. Voir figure ci-dessous :

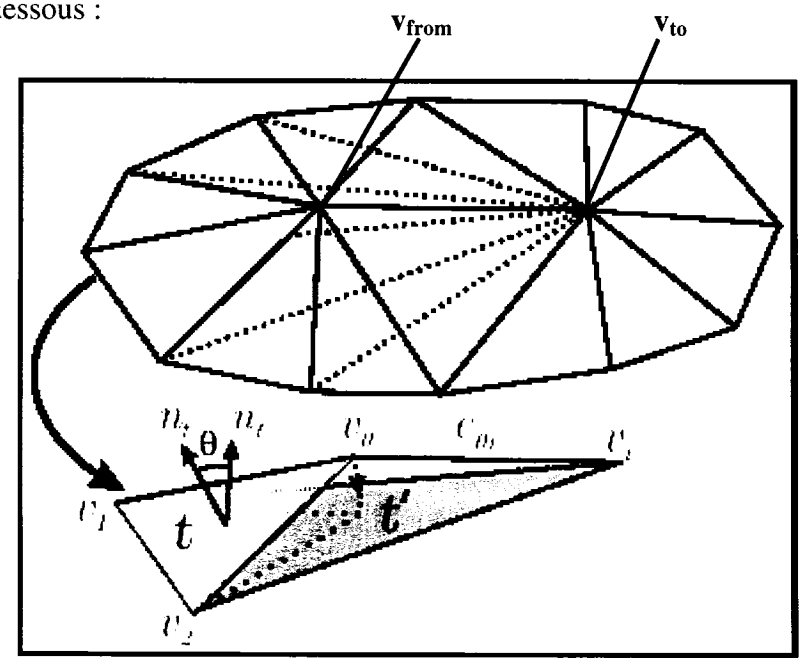



FIGURE 4.2 –Opération de contraction sur les arêtes intérieures

L'angle de rotation décrit plus haut pondéré par la moyenne des superficies, anciennes et nouvelles, représente l'erreur géométrique pour un triangle :

$$\mathbf{Q_t = l_t * \theta_t}$$

où $\mathbf{l_t = 0{,}5(\Delta_t + \Delta_{t'})}$ représente la moyenne de la somme des anciennes et nouvelles superficies.

Dans l'implémentation, on approche l'angle $\theta_t$ par $1-\vec{n}_t \bullet \vec{n}_{t'}$ où $\vec{n}_t$ et $\vec{n}_{t'}$ sont les vecteurs normaux des triangles $t$ et $t'$. Cette approximation répond bien aux besoins de l'algorithme et épargne beaucoup de temps au niveau des calculs étant donné l'élimination des opérations d'évaluation d'angles via les fonctions trigonométriques.

Néanmoins, il est à remarquer qu'il ne s'agit que d'une approximation grossière de $\theta_t$. Nous assumons que la valeur de $\theta_t$ peut varier de $0^\circ$ à $180^\circ$ ou encore entre 0 et $\pi$ radians. En normalisant la variation d'angle par $\frac{1}{\pi}$, nous obtenons la variation entre 0 et 1. La variation de la fonction $\theta_t = 1-\vec{n}_t \bullet \vec{n}_{t'}$ quant à elle varie entre 0 et 2 mais en normalisant cette dernière par $\frac{1}{2}$ nous obtenons les mêmes bornes de variation entre 0 et 1.

Maintenant, comparons la variation des valeurs exactes d'angle $\theta$ entre les deux vecteurs normalisés $\vec{a}$ et $\vec{b}$ et son approximation $1-\vec{a} \bullet \vec{b}$. Pour ce faire, nous fixons le vecteur $\vec{b}$ à une valeur $\vec{b}=(1,0)$ et varierons le vecteur $\vec{a}$ de (-1,0) à (1,0) dans le sens des aiguilles d'une horloge.

Les variations de $\theta = \arccos(\vec{a} \bullet \vec{b})$ et $\theta = 1-\vec{a} \bullet \vec{b}$ normalisées sont représentées sur la figure ci-dessous :

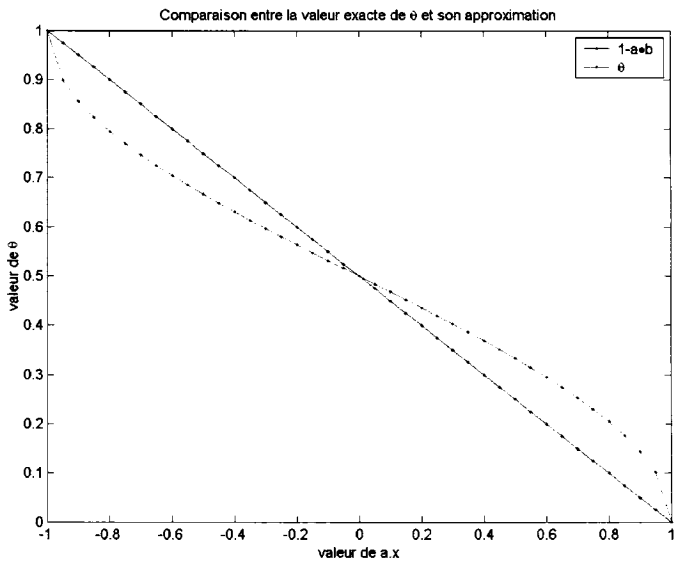



FIGURE 4.3 – Comparaison entre la valeur exacte d'un angle et son approximation

Comme on peut le constater, l'écart entre la valeur exacte de $\theta$ et son approximation est de l'ordre de 10 % de la valeur maximale. Notre but est de comparer les erreurs causées par les différentes contractions des arêtes et pour les contractions pratiquement faisables, la valeur de $\theta$ est très inférieure à $90^{\circ}$; donc toutes les arêtes se trouvant dans la même plage de valeurs causeront les mêmes erreurs. Comme l'approximation d'angle est monotone, la différence dans l'estimation n'affectera que très peu les résultats de comparaison des coûts de suppression.

Finalement, le coût de suppression d'une arête « $e(v_{from}, v_{to})$ » est égal à la somme des erreurs géométriques calculées pour chacun des triangles voisins de « $v_{from}$ ».

$$\text{Coût } e(v_{from}, v_{to}) = \sum Q_t$$

où « t » appartient à l'ensemble des triangles adjacents au sommet « $v_{from}$ » à l'exception de deux triangles adjacents à l'arête « $e(v_{from}, v_{to})$ » .

### 4.2.3 Traitement des frontières

Les arêtes dont les sommets sont situés sur les frontières sont traitées différemment de celles à l'intérieur de la surface. Tout d'abord, les arêtes frontalières se divisent en deux catégories :

1. Arêtes avec un sommet sur la frontière et un sommet à l'intérieur de la surface
2. Arêtes avec deux sommets sur la frontière

#### 4.2.3.1 Traitement des arêtes avec un sommet sur la frontière

Si le sommet $v_{from}$ est situé sur la frontière, alors la suppression de cette arête peut causer une déformation sévère de ladite frontière. Par conséquent, la suppression d'une arête « $e(v_{from}, v_{to})$ » où $v_{from}$ est situé sur la frontière est prohibée.

Le cas d'une arête dont $v_{to}$ est situé sur la frontière ne requiert pas de traitement spécial lorsque le sommet qui doit être supprimé ($v_{from}$) est situé à l'intérieur de la surface.

#### 4.2.3.2 Traitement des arêtes avec deux sommets sur la frontière

Les arêtes de la deuxième catégorie où $v_{from}$ et $v_{to}$ sont situés sur la frontière sont traitées de la façon suivante :

Le coût de suppression d'une telle arête est pondéré par la longueur de celle-ci et par l'angle entre l'ancienne et la nouvelle arête. Voir figure ci-dessous :

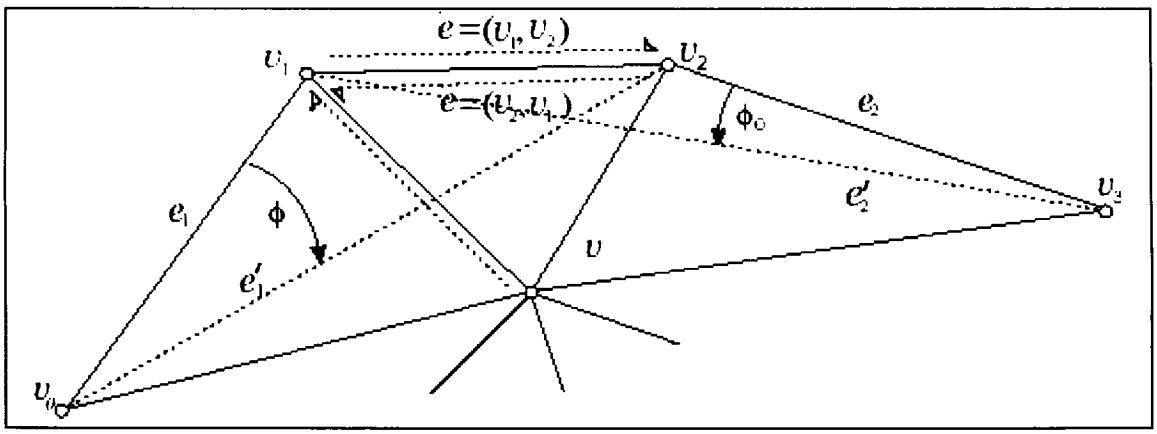



FIGURE 4.2 –Opération de contraction sur les arêtes frontalières

À remarquer sur la figure que l'arête « $e(v_2, v_1)$ » peut être contractée au sommet $v_1$ ou $v_2$ mais pour parvenir au meilleur résultat, elle doit être contractée au sommet $v_1$.

Pour ce faire, le coût de suppression des arêtes « $e(v_2,v_1)$ » et « $e(v_1,v_2)$ » doit être multiplié par $\varphi$ et $\varphi_0$ respectivement. En résumé, le coût de suppression d'une arête dont les sommets sont situés sur la frontière sera :

$$\text{Coût } e(v_1, v_2) = \lambda\phi\|v_1 - v_2\| + \sum_{t \in Tv1 - Tv2} Qt$$

où $\lambda$ est un poids de conservation des frontières.

La valeur de $\lambda$ peut varier dépendamment de la qualité désirée lors de la préservation des frontières. Dans notre implémentation la valeur de $\lambda$ a été fixée à 50 ; avec cette valeur, les frontières demeurent intactes lors de la simplification, tel que déterminé par l'auteur de l'algorithme M. Hussain.

### 4.2.4 Description sommaire de l'algorithme de M. Hussain

**Étape 1 :**

1. Pour chaque sommet $v_i$ :
   a) Calculer son importance visuelle
2. Pour chaque sommet voisin $v_j$ dans le voisinage ( $Nv_i$ ) de $v_i$ :
   a) Calculer le coût de suppression $c_{ij}$ de chaque arête « $e(v_i,v_j)$ » où $v_j \in Nv_i$ en utilisant une des formules de coût décrites ci-dessus (dépendamment du genre d'arête, soit frontalière ou à l'intérieur de la surface)
   b) Déterminer « $e(v_i,v_j)$ » avec le coût minimal ie $c_{imin} = \min(c_{ij}$ ou $v_j \in Nv_i)$
3. Pondérer $c_{imin}$ avec l'importance visuelle de $v_i$ et le mettre dans la file triée selon l'ordre croissant des valeurs. Associer au sommet $v_i$ le sommet $v_j$ avec le coût de contraction minimum.

**Étape 2 :**

4. Pour chaque arête dans la file des arêtes :
   a) Enlever de la file le sommet $v_i$ au coût inférieur

b) Supprimer l'arête « $e(v_i,v_j)$ ». Pour ce faire, substituer toutes les occurrences de $v_i$ par $v_j$ et supprimer les triangles adjacents à l'arête « $e(v_i,v_j)$ ».

c) Lorsque nous avons changé le voisinage de $v_j$, pour chaque sommet $v_k \in Nv_i$ mettre à jour le coût de suppression tel que décrit à l'étape 1.

Répéter l'étape 2 tant qu'il reste des sommets dans la file.

## 4.3 3e algorithme - Simplification des surfaces via la norme de courbure discrète

Cet algorithme est basé sur l'évaluation de la norme de courbure discrète. Cette évaluation, contrairement à sa version continue ne requiert pas la continuité des surfaces et peut être appliquée aux maillages. Elle est basée sur un raisonnement géométrique décrit dans la section suivante.

### 4.3.1 Courbure normale

La courbure normale $K_n$ de la surface est la courbure d'une courbe appartenant à la surface. Cette courbe est le résultat de l'intersection d'un plan (qui contient le vecteur normal de la surface) et de la surface elle-même, en un point donné de la surface.

### 4.3.2 Courbures principales

Les courbures principales $k_1$ et $k_2$ sont les maximum et minimum de la courbure normale en un point donné de la surface. (*voir figure 4.4*) :

FIGURE 4.4 – Partie de la surface avec le minimum et le maximum de la courbure normale

### 4.3.3 Courbure Gaussienne

La courbure Gaussienne est une propriété locale d'une surface qui ne dépend que de la géométrie intrinsèque de cette surface. Cette propriété est basée sur le fait que la somme des angles intérieurs d'un polygone sur la surface courbée est supérieure ou inférieure à $360^{o}$. Par exemple, si nous mesurons les angles d'un triangle sur une sphère, la somme des angles sera supérieure à $180^{o}$. Plus explicitement, lorsque la courbure est positive la géométrie locale aura l'apparence d'une bosse tandis que si elle est négative, sa géométrie aura une apparence concave. Évidemment, si la courbure est égale à 0, alors la géométrie locale est un plan. Si nous additionnons (ou intégrons) la courbure Gaussienne de tous les points de la surface, nous obtenons la courbure totale.

Une formulation simple du théorème de Gauss-Bonnet stipule que la courbure totale d'une surface fermée est un invariant :

$$CourbTot=2\pi\ (2-2T)$$

où T représente le nombre de trous sur la surface.

Dans le cas d'une surface avec une frontière, la déformation de cette surface n'affecte pas la courbure totale de celle-ci si elle ne modifie pas la courbure aux bords. (Guilbault, F., 2005).

Notons par K (p) la courbure Gaussienne au point p. Appliqué à une région engendrée par des courbes lisses, le théorème de Gauss-Bonnet affirme que :

$$\iint_D K(p)dA+\sum\int_{\Gamma_j} k_g(\Gamma_j)ds+\sum\alpha_j=2\pi$$

où $k_g$ est la courbure géodésique de la courbe frontalière et $\alpha_j$ est l'angle extérieur au point $p_j$ de la frontière. (voir figure ci-dessous) :

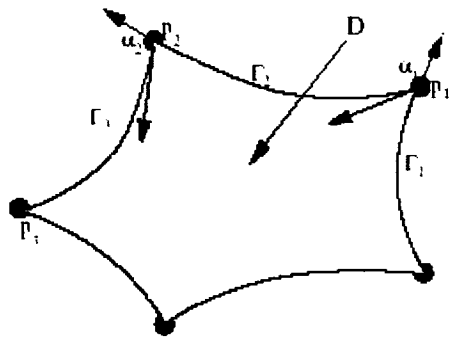



FIGURE 4.5 – Région de la surface (D) engendrée par les courbes frontalières $\Gamma_j$

Cependant, si toutes les frontières $\Gamma_j$ sont les courbes géodésiques, la formule ci-haut est réduite à :

$$\iint_D K(p)dA=2\pi-\sum\alpha_j$$

Maintenant considérons la discrétisation de $K(p)$ sur la triangulation d'une surface continue.

En assumant que $K(p)$ est distribué uniformément dans le voisinage local du sommet p, l'équation précédente est discrétisée comme suit :

$$K(p)=\frac{3(2\pi-\sum\theta_i)}{A(p)}$$ (Do Carmo, M.P., 1976)

Il s'agit de la formule que nous avons utilisée en qualité d'opérateur de la courbure Gaussienne, où $A(p)$ représente la somme des superficies des triangles voisins et $\theta_i$ est l'angle intérieur du triangle (voisin) i.

Puisque l'opération d'évaluation de l'angle $\theta_i$ se produit plusieurs fois pour chaque sommet du maillage, le temps d'exécution de l'implémentation, en utilisant la fonction arccos de la librairie math.h, ne fut aucunement performant étant une centaine de fois plus lent que le temps d'exécution des deux premiers algorithmes. Puisqu'une telle implémentation s'avérait trop coûteuse au niveau du temps dont nous disposions, nous avons préféré recourir à l'approximation de la fonction arccos afin d'économiser les ressources.

Dans la section 4.2.2, nous avons mentionné le remplacement d'une fonction trigonométrique arccos par son approximation. Dans le présent cas, il n'était pas possible d'utiliser la même approximation pour les raisons mentionnées à la section 4.2.2, soit un grand écart entre la valeur d'un angle et son approximation.

Comme l'approximation de la fonction arccos(x) est elle-même très ardue, nous avons opté pour approcher la fonction plus simple :

$$approx(y)=arccos(1-y)\sqrt{2y}$$

Avec un changement de variable y=1−x cette approximation devient:

$$\text{approx}(1-x)=\arccos(x)\sqrt{2-2x}\,.$$

que l'on peut également écrire comme suit :

$$\arccos(x)=\frac{\text{approx}(1-x)}{\sqrt{2-2x}}$$

Cette fonction est presque linéaire et peut être approchée sur un intervalle où x varie de 0 à 1 à l'aide du polynôme d'approximation de degré 5 dont les constantes ont été obtenues par la méthode des moindres carrés :

$$\begin{aligned}\text{approx}(1-x) = {} & -0.000007239283986332 + \\ & 2.000291665285952400 * (1-x) + \\ & 0.163910606547823220 * (1-x)^2 + \\ & 0.047654245891495528 * (1-x)^3 - \\ & 0.005516443930088506 * (1-x)^4 + \\ & 0.015098965761299077 * (1-x)^5\end{aligned}$$

Seules cinq opérations de multiplications et d'additions sont requises pour implémenter la fonction, ce qui constitue un net avantage.

Comme on peut le noter sur la figure 4.6, l'écart entre les deux fonctions est minime. L'approximation se trouvant dans la partie supérieure, pour un intervalle de −1 à 0 a été obtenue par l'effet miroir. i.e. que si la valeur de x est négative, nous déterminons la valeur d'approximation pour une valeur de x positive et l'approximation finale est égale à $\pi$ − l'approximation pour un x positif.

La figure ci-dessous représente graphiquement une comparaison entre la fonction arccos exacte et son approximation :

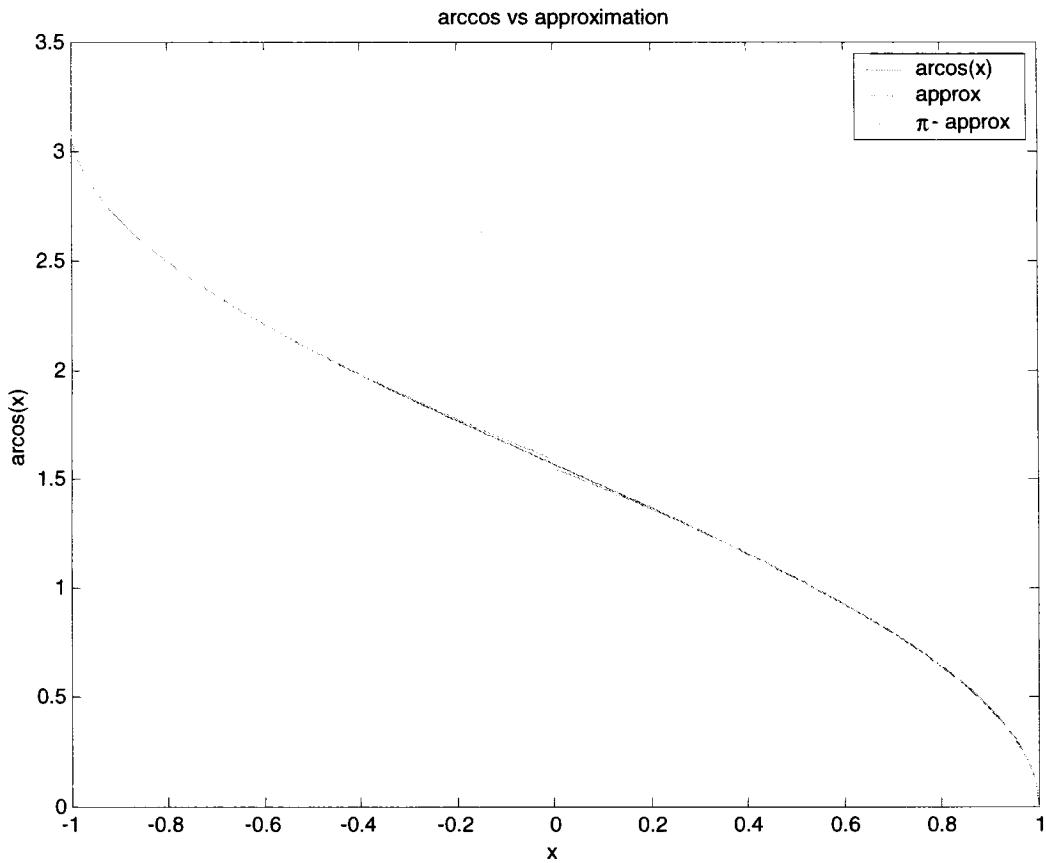



FIGURE 4.6 –Comparaison de la fonction arccos(x) et son approximation

### 4.3.4 Courbure moyenne

La courbure moyenne H est définie comme la moyenne de toutes les courbures normales en un point donné de la surface :

$$H=\frac{1}{2\pi}\int_0^{2\pi} K_n(\theta)d\theta$$

En exprimant la courbure normale $K_n$ par les courbures principales $k_1$ et $k_2$ on obtient la formule :

$$K_n=k_1\cos^2\theta+k_2\sin^2\theta$$

Ce qui amène l'expression bien connue :

$$H=\frac{(k_1+k_2)}{2}$$

Notons par H(x) l'opérateur discret de la courbure moyenne du sommet x et par A la région du voisinage du sommet x.

Maintenant, nous voulons calculer l'intégrale de la courbure moyenne sur la région A. Puisque l'opérateur de la courbure moyenne est également connu comme l'opérateur Laplace-Beltrami, nous calculons un Laplacien de la surface dans l'espace paramétrique avec les paramètres **u** et **v**. Pour ce faire, nous utilisons la discrétisation de la surface comme un espace paramétrique conforme; i.e. que pour chaque triangle du maillage, c'est le triangle lui-même qui définit la métrique locale de la surface.

Alors l'opérateur de Laplace-Beltrami se transforme en simple Laplacien $\Delta_{u,v}x=x_{uu}+x_{vv}$ :

$$\iint_A H(x)dA=-\iint_A \Delta_{u,v}x\,du\,dv$$

En utilisant le théorème de Gauss, on peut démontrer que l'intégrale du Laplacien sur la surface est réduite à la formule simplifiée (Meyer, M. et al., 2002) :

$$\iint_A H(x_i)dA=\frac{1}{2}\sum_{j\in Nx_i}(\cot\alpha_{ij}+\cot\beta_{ij})(x_i-x_j) \quad (1)$$

où $\alpha_{ij}$ et $\beta_{ij}$ sont les deux angles opposés à l'arête $(x_i, x_j)$ (voir figure 4.7)

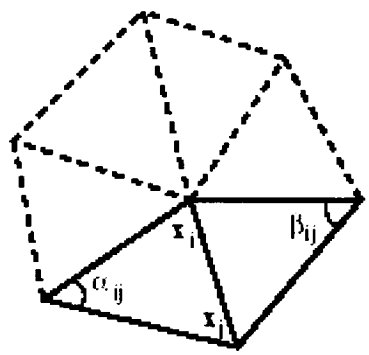



FIGURE 4.7 – Calcul de courbure moyenne discrète

Dans l'algorithme « Simplification de surfaces via l'erreur de courbure discrète » (Kim, S.–J. et al., 2002) la courbure moyenne H est obtenue par l'équation :

$$\iint_A H(x_i)dA = \frac{1}{4}\sum_{j \in Nx_i} \beta_{ij}(x_i - x_j)$$

où $\beta_{ij}$ est un angle entre deux normales des triangles adjacents, ou l'angle dihedral, de l'arête $\vec{e}_{ij}$ .

Cette façon d'évaluer la courbure moyenne s'avère inefficace lors de l'implémentation de l'algorithme puisqu'elle demande l'évaluation de l'angle $\beta_{ij}$ via les fonctions trigonométriques *arccos* et *arcsin.* Pour cette raison, nous avons changé l'opérateur de courbure moyenne décrite dans l'article (Kim, S.–J. et al., 2002) par celui dérivé de la formule (1) :

$$H(x_i) = \frac{3}{2A}\sum_{j \in Nx_i} (\cot \alpha_{ij} + \cot \beta_{ij})(x_i - x_j)$$ (Meyer, M. et al., 2002)

Cet opérateur est plus efficace puisque deux évaluations de *cot* pour chaque triangle demande approximativement dix fois moins de temps qu'une évaluation de *arccos.*

### 4.3.5 Métrique d'erreur basée sur l'évaluation de la courbure discrète

Les courbures Gaussiennes et moyenne sont reliées aux courbures principales par les équations suivantes : $K = k_1 k_2$ et $H = \frac{k_1 + k_2}{2}$,

ce qui peut être écrit comme une équation quadratique : $k^2 - 2Hk + K = 0$

Les solutions sont donc :

$$k_1 = H + \sqrt{H^2 - K}$$

$$k_1 = H - \sqrt{H^2 - K}$$

Maintenant, nous pouvons définir la norme de la courbure discrète R pour chaque sommet :

$$R = |k_1| + |k_2|$$

Par substitution, nous obtenons :

$$|k_1| + |k_2| = \begin{cases} 2H & \text{si } K \geq 0 \\ 2\sqrt{H^2 - K} & \text{autrement} \end{cases}$$

Il faut noter que dans le cas des surfaces discrètes, il peut arriver que la valeur de K soit supérieure à celle de $H^2$, alors dans ce cas nous poserons la valeur de K à 0. Durant la simplification, la norme de la courbure discrète R est calculée avant et après la fusion des sommets. La différence entre ces deux valeurs indique la différence d'apparence avant et après la simplification et joue le rôle de métrique d'erreur. Comme dans les algorithmes précédents, nous choisissons en premier une arête qui minimise la différence entre les normes des courbures discrètes avant et après la fusion des sommets.

### 4.3.6 Description de l'algorithme basé sur l'évaluation de la norme de la courbure discrète

**Étape 1 :**

1. Pour chaque sommet $v_i$ du maillage :
2. Pour chaque sommet voisin $v_j$ dans le voisinage de $v_i$, (*trouver l'arête issue de $v_i$ dont la suppression minimise la variation de courbure*):
   a) Calculer la norme de courbure discrète de $v_j$ avant la fusion
   b) Simuler la fusion en remplaçant toutes les occurrences de $v_i$ par $v_j$
   Puisque le voisinage de $v_j$ est changé, le mettre à jour.
   c) Calculer la norme de courbure discrète de $v_j$ après la fusion

d) Calculer la variance des normes de courbures discrètes avant et après la fusion

*Si la variance du sommet* $v_j$ *est inférieure à celle de* $v_{j-1}$ *(voisin précédent),alors variance min = variance du sommet* $v_i$ *et l'arête candidate à la suppression est « e(v,*$v_i$*) »*

*Sinon, passer au sommet voisin suivant* $v_{j+1}$

3. Dès que tous les voisins de $v_i$ sont parcourus, mettre l'arête « e( $v_i$ , $v_j$ ) » dont le coût de suppression est minimal dans la file.

**Étape 2 :**

4. Pour chaque arête dans la file des arêtes :
   a) Enlever l'arête au coût inférieur
   b) Supprimer l'arête « e( $v_i$ , $v_j$ ) ». Pour ce faire, substituer toutes les occurrences de $v_i$ par $v_j$ et supprimer les triangles $T \in e_{ij}$
   c) Lorsque nous avons changé le voisinage de $v_j$, recalculer le coût de suppression pour tous les voisins de $v_j$ et ainsi déterminer le nouveau « e( $v_j$ , $v_k$ ) » ayant le coût de suppression minimal.
   d) Mettre à jour la file des arêtes candidates à la suppression.

# CHAPITRE 5

# MÉTHODOLOGIE DE COMPARAISON DES ALGORITHMES

Dans les chapitres précédents nous avons décrit trois algorithmes de simplification de surfaces, notamment la « Simplification des surfaces via l'erreur quadratique » (M. Garland), la « Simplification des surfaces via l'erreur géométrique » (M. Hussain) et enfin la « Simplification des surfaces via l'erreur de courbure discrète » (S.–J. Kim).

Le présent chapitre servira à décrire les démarches entreprises afin de comparer ces algorithmes. Nous élaborerons entre autres sur la description des modèles, leur provenance et leur utilité mais également sur la méthode de récupération de la plupart des modèles. Nous poursuivons par la suite avec la description des cas tests.

## 5.1 Description des modèles

Afin d'effectuer les cas tests adéquats au plan statistique, nous nous sommes construit une bibliothèque de modèles, que nous avons divisée en trois parties; soit :

1. Surfaces pleines représentant les parties des pièces d'écoulement dans le domaine hydroélectrique (99 surfaces);
2. Surfaces plus ou moins courbées, de forme complexe, avec trous ; certaines de ces surfaces proviennent du domaine hydroélectrique alors que d'autres ont été recueillies via Internet et proviennent de d'autres domaines de la science (8 surfaces);
3. Maillages complexes (3 maillages).

### 5.1.1 Surfaces pleines représentant les parties des pièces d'écoulement

Les surfaces pleines, sans trous, représentant en partie des pièces d'écoulement, nous ont été fournies par la compagnie GE Hydro. Ces surfaces font partie des pièces d'écoulement appelées « aspirateurs » lesquels peuvent être de trois types :

1. Sans pile
2. À une pile
3. À deux piles

Voir figure 5.1 page 60.

Quant à la méthode de récupération des surfaces, celle-ci sera décrite plus explicitement à la section suivante.
Voir figure 5.2 page 60 pour trois exemples de surfaces de la première catégorie dont nous venons de parler.

### 5.1.2 Surfaces de forme complexe, avec trous

Les surfaces avec trous et les surfaces avec frontières complexes ont été recueillies à partir de différentes sources. En effet, l'une des surfaces provient du domaine hydroélectrique alors que certaines autres ont été recueillies via Internet. Voir la figure 5.3, page 60 pour la représentation graphique de trois surfaces avec trous.

### 5.1.3 Maillages complexes

Enfin, les modèles complets ont été maillés à partir de descriptions géométriques, lesquelles ont été écrites en format .nurbs. Ce format suppose la description d'un modèle utilisant uniquement des données paramétriques de surfaces NURBS et de Bézier.
La façon de décrire paramétriquement les surfaces en format .nurbs est identique à la description du format .pie qui a été élaboré par un groupe de chercheurs issus de l'École Polytechnique de Montréal. Le format .pie est un format standard d'entrées/sorties des logiciels développés au laboratoire MAGNU (http://www.magnu.polymtl.ca).Voir ci-dessous une description formelle du format .nurbs :

NOMBRE DE SURFACES

**Pour chaque surface :**

1. Degré de la surface dans la direction u
2. Degré de la surface dans la direction v
3. Nombre de nœuds dans la direction u
4. Vecteur nodal dans la direction u
5. Nombre de nœuds dans la direction v
6. Vecteur nodal dans la direction v
7. Nombre de points de contrôle dans la direction u
8. Nombre de points de contrôle dans la direction v
9. Tableau des points de contrôle
10. Nombre de boucles de rognage

**Pour chaque boucle :**

1. Nombre de segments

**Pour chaque segment :**

1. Degré de la courbe
2. Type de la courbe (nurbs ou linéaire)
3. Nombre de nœuds du vecteur nodal (seulement si nurbs)

4. Vecteur nodal (seulement si nurbs)
5. Nombre de points de contrôle
6. Tableau de points de contrôle

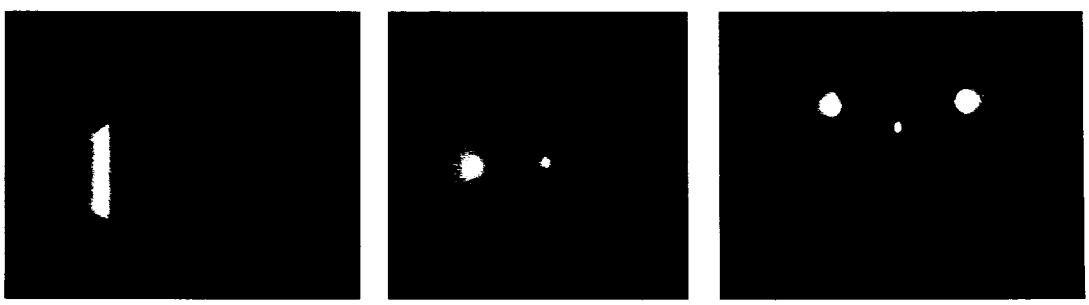
Voir figure 5.4 page 61 pour la représentation graphique de trois modèles dont les maillages ont été récupérés à partir de la description .nurbs .

**Aspirateur sans pile** **Aspirateur à une pile** **Aspirateur à deux piles**

FIGURE 5.1 – Trois types d'aspirateurs

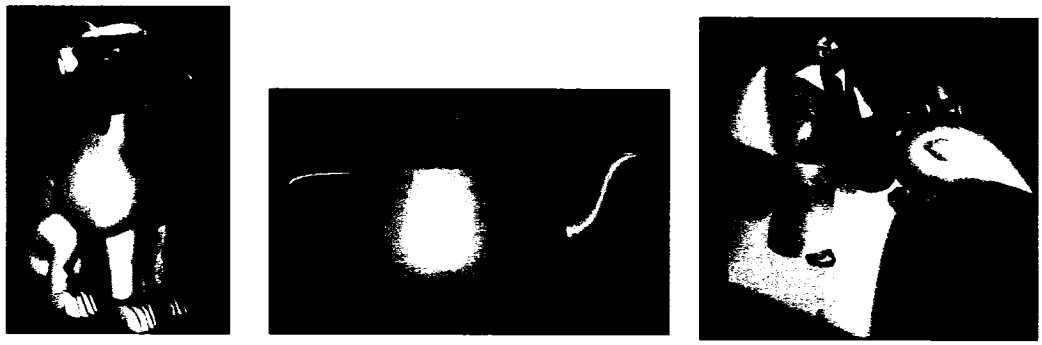
**Partie basse** **Partie droite** **Partie haute**

FIGURE 5.2 – Parties de « l'aspirateur sans pile » de la figure 5.1



**Partie d'aspirateur** **Surface à un trou** **Surface à deux trous**

FIGURE 5.3 – Surfaces complexes avec trous.



**lion.nurbs** **teapot.bezier** **rider.nurbs**

FIGURE 5.4 – Modèles récupérés à partir de la description .nurbs.

## 5.2 Démarche de récupération des maillages à partir de l'interface NURBS d'OpenGL

Les 99 surfaces représentant des pièces d'écoulement ont été recueillies via le logiciel Topovisu que nous avons dû modifier afin de récupérer les informations géométriques sous forme de maillages. La récupération des maillages s'est déroulée de la façon suivante :

Le logiciel Topovisu lit les descriptions paramétriques des surfaces et les informations topologiques associées aux faces dans un fichier .pie. Par la suite, les informations paramétriques, telles que le degré de la surface, les vecteurs nodaux et les points de contrôle sont passés à l'interface NURBS d'OpenGL. Par défaut le fragmenteur NURBS divise un objet NURBS en lignes et polygones géométriques puis les restitue. Dans GLU 1.3, des fonctions de *rappel* supplémentaires ont été ajoutées, ce qui permet de récupérer les valeurs de post-fragmentation. Ces valeurs peuvent être retournées afin d'être utilisées par l'application. Pour ce faire, la première étape consiste à indiquer que l'interface *rappel* est activée. La deuxième étape consiste à appeler plusieurs fois la fonction *gluNurbsCallback*( ) pour enregistrer les fonctions de *rappel*.

Pendant la fragmentation, les fonctions de rappel fonctionnent de façon similaire aux commandes d'OpenGL : *glBegin ( ), glVertex ( )* et *glEnd ( ).*

Lorsqu'elle est restituée en mode GLU_NURBS_TESSELATOR, la courbe ou la surface n'est pas directement restituée mais nous pouvons capturer les données des sommets passés comme paramètres aux fonctions de *rappel.*

Nous présenterons les détails de l'architecture et de l'implémentation dans l'annexe A du présent travail.

## 5.3 Description des cas tests servant à la comparaison des algorithmes

Dans la présente section, nous décrirons en détail les cas tests que nous avons élaborés afin d'effectuer la comparaison des algorithmes. Cette partie de notre travail s'avère être la plus importante puisqu'elle représente un élément nouveau dans la méthodologie de comparaison des algorithmes.

### 5.3.1 Premier cas test – Comparaison selon la distance d'Hausdorff

Les surfaces décrites à la section 5.1.1, au nombre de 99, récupérées à l'aide de l'interface NURBS ont été simplifiées selon neuf niveaux de réduction des triangles en ayant recours à chacune des trois méthodes suivantes :

1. Simplification via l'erreur quadratique avec pondération par la superficie des triangles (M. Garland)
2. Simplification via l'erreur géométrique (M. Hussain)
3. Simplification via la norme de courbure principale dont l'algorithme est basé sur l'article de S.-J. Kim

Les niveaux de réduction des triangles sont de 95 %, 90 %, 80 %, 70 %, 60 %, 50 %, 40 %, 30 %, 20 % et de 10 % du nombre initial de triangles. Par exemple, pour un niveau de réduction de 95 %, il ne reste que 5% de triangles du nombre initial. Par la suite, les maillages réduits sont comparés aux maillages initiaux selon la métrique d'erreurs d'Hausdorff.

Les résultats d'erreurs tels $E_{max}$, $E_{min}$, $E_{moy}$ et $E_{qmoy}$, ont été enregistrés pour chacun des maillages comparés.
Nous avons par la suite cumulé les résultats d'erreurs pour chacune des méthodes afin de trouver la moyenne de chacune de ces erreurs ; les résultats finaux ont été tracés sous forme de quatre graphiques (erreur versus pourcentage de triangles restants), chacun représentant un type d'erreurs. Ces graphiques sont représentés à la section 6.3.1.

### 5.3.2 Comparaison selon le temps d'exécution

Premièrement nous déterminerons la complexité asymptotique des différents algorithmes à la section 6.1.1. Par la suite, concernant le temps d'exécution empirique, nous le mesurerons lors de la simplification des 99 surfaces mentionnées à la section 5.1. Le temps de la simplification pour un taux de réduction de triangles à 100 % sera considéré étant donné la difficulté d'estimer le temps de la simplification d'une partie de la surface.. Par la suite, nous regrouperons les modèles selon leur taille et pour chaque groupe, nous cumulerons les temps de simplification. Les résultats seront représentés aux sections 6.1.3.1 et 6.1.3.2.

### 5.3.3 Deuxième cas test - Comparaison selon la conservation des frontières

Ces comparaisons sont effectuées dans le but d'évaluer la meilleure méthode de conservation des frontières extérieures et intérieures des surfaces. Ce test est appliqué aux surfaces décrites à la section 4.1.2. En tout, nous avons récupéré huit (8) surfaces.

Les surfaces avec trous ont été simplifiées comme pour le cas précédent, soit avec neuf niveaux de réduction et ce, en ayant toujours recours aux trois méthodes mentionnées précédemment. Nous avons examiné visuellement les maillages simplifiés afin d'évaluer laquelle des trois méthodes performe le mieux en terme de préservation des trous et des frontières des surfaces initiales. Nous avons également effectué la comparaison selon la distance d'Hausdorff, de façon semblable à celle décrite à la section précédente et ainsi tracé des graphiques, lesquels sont représentés à la section 6.1.3.2.

Nous avons également procédé à la comparaison des temps d'exécution et les résultats de ces comparaisons se trouvent à la section 6.1.3.2.

### 5.3.4 Troisième cas test – Comparaison selon la conservation des détails

Les maillages complexes décrits à la section 4.1.3 furent simplifiés toujours selon les neuf niveaux de réduction déjà énumérés ci-haut. Nous avons aussi examiné visuellement les maillages simplifiés afin d'évaluer laquelle des trois méthodes performe le mieux en terme de préservation du niveau de détail initial. Ces résultats sont représentés à la section 6.1.1.3.

# CHAPITRE 6

# RÉSULTATS ET ANALYSE

Dans les chapitres précédents, nous avons décrits les algorithmes de simplification des surfaces (chapitre 4) et cas tests élaborés afin de recueillir les résultats statistiques de simplification (chapitre5, section 5.3). Le présent chapitre sera dédié à l'analyse de la qualité des résultats produits par les trois algorithmes afin de comparer leur efficacité. Nous conclurons par un résumé des comparaisons des algorithmes ainsi qu'avec des recommandations.

## 6.1 Comparaison selon temps d'exécution et consommation de mémoire

La majorité des comparaisons seront basées sur les résultats empiriques mais nous commencerons quand même avec quelques analyses théoriques quant à la complexité asymptotique des algorithmes ainsi que de leur efficacité en terme de consommation de mémoire

### 6.1.1 Analyse de complexité asymptotique

La simplification décrite précédemment au chapitre 4 pour les trois algorithmes peut être divisée en deux phases distinctes. La première phase est une initialisation dans laquelle les paires de candidats pour la simplification sont sélectionnés et placés dans une file selon le coût de simplification. La complexité asymptotique de cette partie pour les trois algorithmes est $O(n\log n)$ où $n$ représente le nombre de sommets du maillage. Puisque les trois algorithmes sont voraces, à chaque itération un candidat est éliminé.

La deuxième partie consiste en la simplification elle-même. À cette étape, la file de candidats est parcourue, la paire de candidats au coût minimum est sélectionnée, la fusion des sommets est effectuée et le voisinage local est mis à jour.

La complexité de cette partie pour les trois algorithmes est :

$$\log n + \log(n-1) + \log(n-2) \ldots + \log m$$

où m est nombre de sommet final.

On peut approcher la complexité de la phase de simplification par $\log n!$ ou encore par $n \log n$ .

Finalement, la complexité asymptotique finale pour les trois algorithmes est $O(n \log n)$

### 6.1.2 Usage de mémoire

Outre le temps d'exécution, la consommation de mémoire est un aspect très important de l'efficacité. C'est particulièrement vrai pour les modèles représentés par un très grand nombre de triangles. Dans cette comparaison, l'algorithme « Simplification des surfaces via l'erreur quadratique » requiert au minimum 128 octets de mémoire par sommet de plus que les autres algorithmes. En plus, selon nos résultats empiriques, nous pouvons affirmer que cet algorithme cause une pagination excessive de la mémoire virtuelle dans le cas des modèles de grande dimension, ce qui corrobore les affirmations de l'auteur dudit algorithme (M. Garland, 1999).

### 6.1.3 Temps d'exécution empirique

À la section 6.1.1 nous avons établi que la complexité des trois algorithmes est $O(n \log n)$. Ceci signifie que le temps d'exécution est borné même pour la simplification de modèles composés d'un grand nombre de triangles. Néanmoins, en pratique, le temps d'exécution réel est plus important.

Dans la présente section, nous ferons la comparaison du temps d'exécution des algorithmes et ce, pour les trois types de surfaces décrits aux sections 5.1.1 et 5.1.2.

Pour la première catégorie de surfaces (section 5.1.1), nous avons regroupé les temps obtenus suite à la simplification de 99 surfaces en huit groupes distincts et ce, en fonction de leur nombre de sommets. L'estimé du temps d'exécution selon la distribution de Student avec un intervalle de confiance de 90 % a été calculé pour chacun des groupes et les résultats sont représentés dans des graphiques pour nos trois algorithmes. Pour les programmes de calculs d'intervalles de Student se référer à l'annexe B.

Un estimé par intervalle est obtenu de la façon suivante :

$$\bar{x} - t_{\alpha/2,n-1}\frac{S}{\sqrt{n}} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}\frac{S}{\sqrt{n}}$$

où $n$ représente la taille de l'échantillon, $\bar{x}=\frac{\sum_{i=1}^{n} x_i}{n}$ est un estimé ponctuel de la moyenne, $t_{\alpha/2,n-1}$ est une valeur obtenue de la distribution de Student avec $n$-1 degré de liberté pour un niveau de risque $\alpha$ (10 %) et S est un estimé sans biais de l'écart type obtenu par la formule :

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Les résultats ont été obtenus sur un système équipé d'un processeur AMD Athlon(™) XP1700+ , dont la cadence est 1,47GHz et la mémoire principale de 512 Mo de RAM.

Ci-dessous représentation de la moyenne du temps d'exécution versus le nombre de sommets simplifiés.

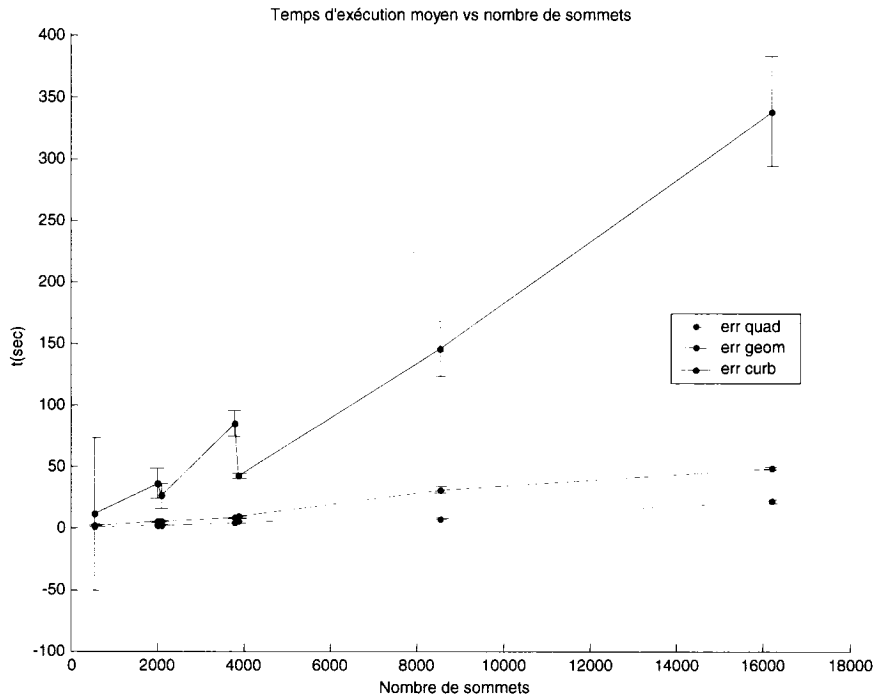



FIGURE 6.1 – Temps d'exécution des trois méthodes pour 99 surfaces d'écoulement

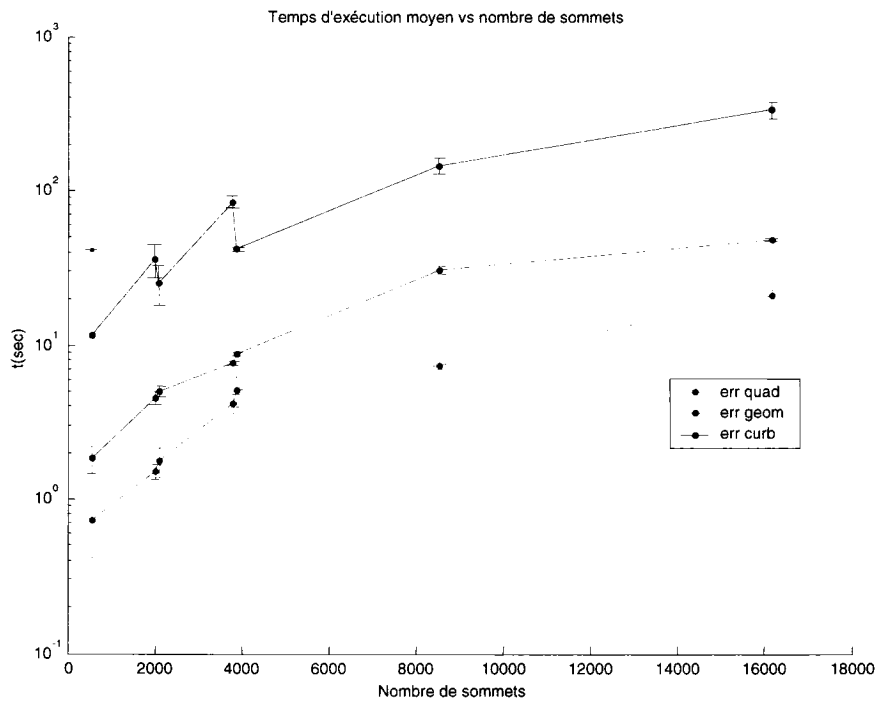



FIGURE 6.2 – Temps d'exécution des trois méthodes pour 99 surfaces d'écoulement en échelle logarithmique.

Les résultats de ces tests sont concluants, surtout étant donné leur caractère statistique quant aux données recueillies. On peut noter que l'algorithme basé sur l'évaluation de l'erreur quadratique performe mieux alors que l'algorithme basé sur l'évaluation de l'erreur géométrique est environ deux fois plus lent, ce qui correspond aux résultats obtenus lors d'études antérieures par d'autres chercheurs (Hussain, M. et al., 2003).

Nous pouvons remarquer que pour les deux premiers algorithmes, le comportement du temps d'exécution est presque linéaire.

Quant au troisième algorithme, son temps d'exécution n'est pas comparable avec les deux premiers et ce, dû au fait qu'il exige beaucoup d'évaluation au niveau des fonctions trigonométriques.

Pour la deuxième catégorie de surfaces (section 5.1.2), nous avons procédé aux mêmes démarches que pour la première catégorie, à la différence que nous n'avons que 8 surfaces au lieu de 99. Ci-dessous nous présentons un graphique du temps d'exécution des trois algorithmes. À noter que les axes sont en échelle logarithmique afin d'en faciliter la visualisation.

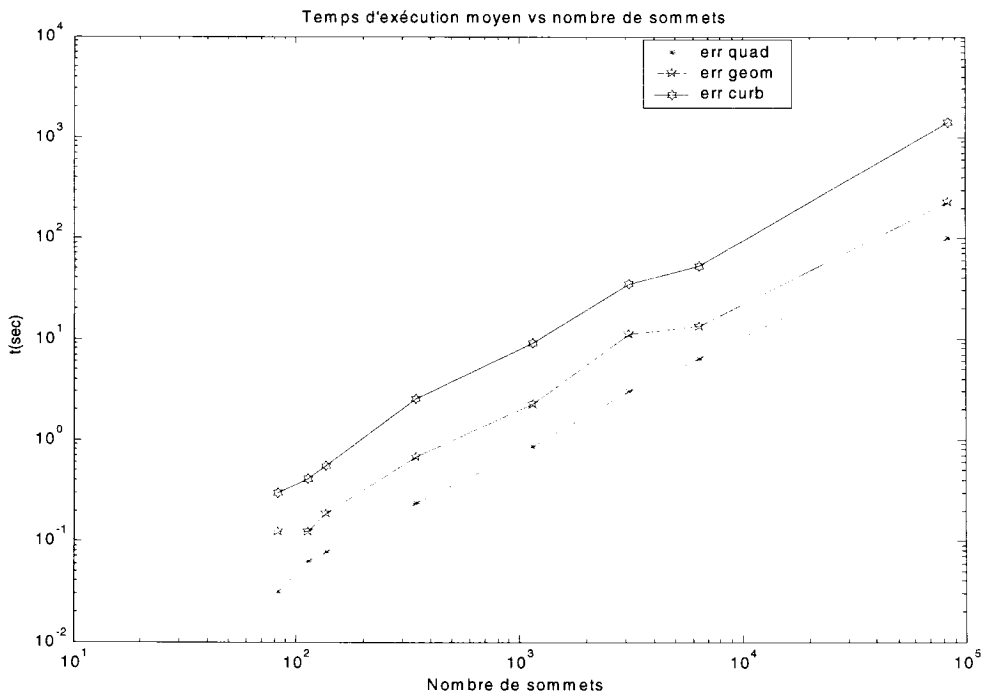



FIGURE 6.3 - Temps d'exécution des trois méthodes pour 8 surfaces avec trous

À remarquer que les résultats démontrés ci-dessus sont très similaires à ceux obtenus lors de la comparaison de la première catégorie de surfaces.

### 6.1.3 Qualité géométrique des résultats

À la section précédente, nous avons procédé à la comparaison des algorithmes selon leur temps d'exécution mais ce critère n'est pas le seul facteur influençant le choix d'un algorithme; l'efficacité du temps d'exécution peut être inutile si l'algorithme n'approche pas le modèle original.

Dans la présente section, nous comparons les algorithmes selon leurs résultats produits en terme de qualité d'approximation. Ainsi, nous fournirons quelques exemples visuels des résultats de simplification des trois algorithmes et des résultats de comparaison en terme de l'erreur géométrique.

Nous débutons par la présentation des résultats statistiques pour la première catégorie de surfaces mentionnée à la section 5.1.1. Par la suite, nous présenterons les résultats pour la deuxième catégorie de surfaces mentionnées à la section 5.1.2, notamment en ce qui a trait aux surfaces avec trous. Finalement, nous terminerons par la présentation des résultats de la troisième catégorie de surfaces mentionnées à la section 5.1.3.

#### 6.1.3.1 Résultats de comparaison des surfaces de la première catégorie

Tel que mentionné au chapitre 5, les surfaces de première catégorie au nombre de 99 ont été simplifiées avec dix niveaux de réduction de triangles. Les six graphiques illustrés aux pages suivantes représentent l'erreur maximale, l'erreur moyenne et l'erreur quadratique moyenne versus le pourcentage de simplification des triangles.

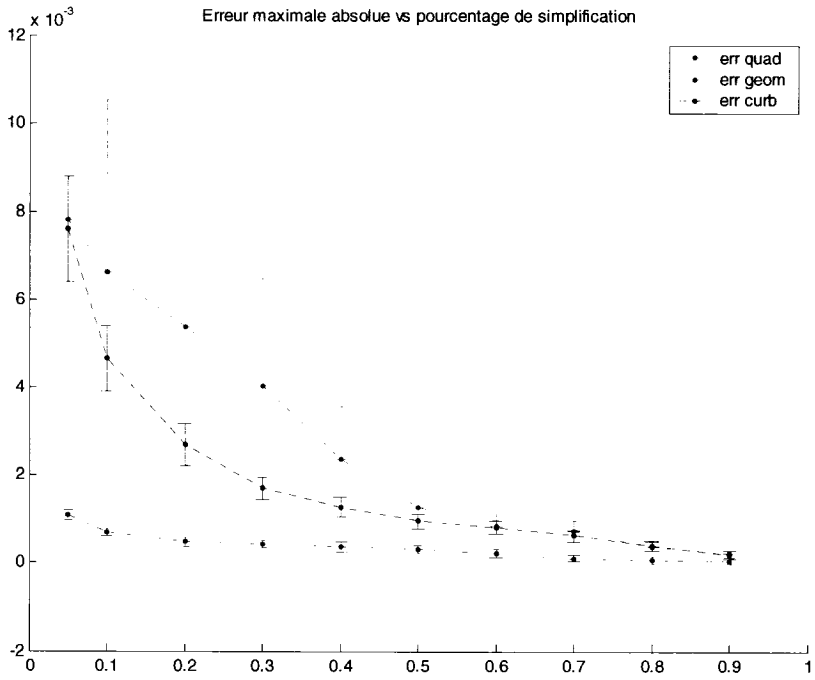



FIGURE 6.4 – Erreur maximale versus pourcentage de simplification

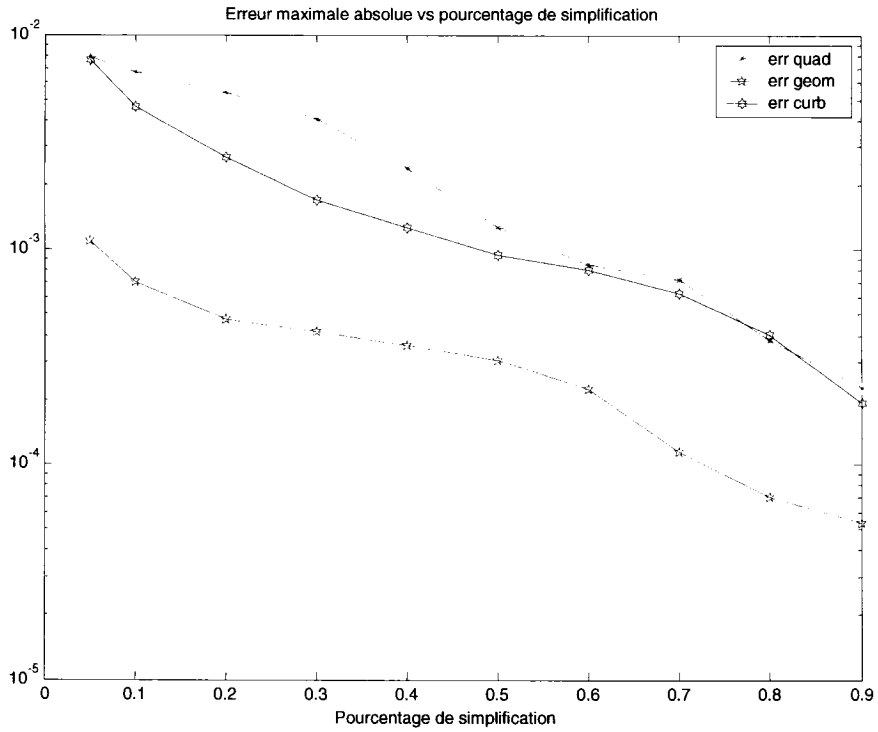



FIGURE 6.5 – Erreur maximale versus pourcentage de simplification en échelle logarithmique

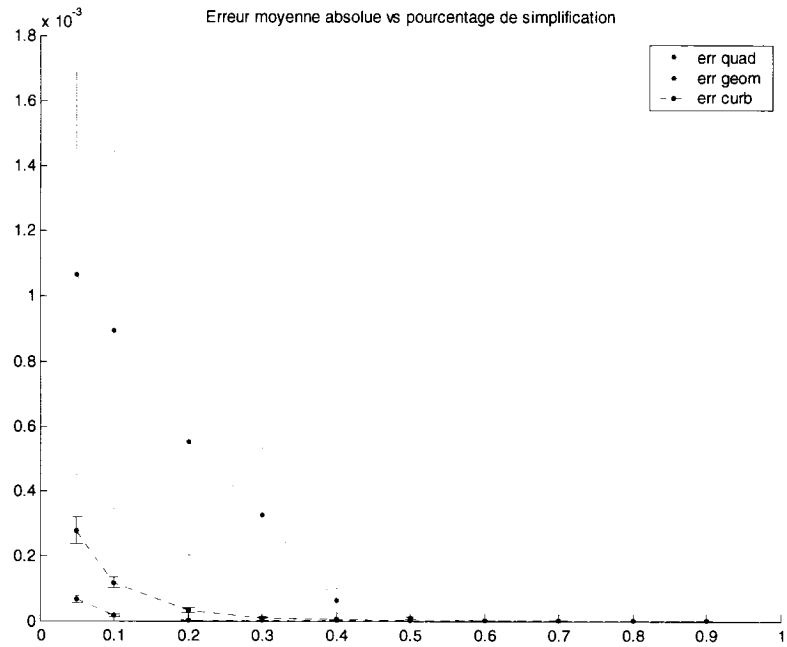



FIGURE 6.6 – Erreur moyenne versus pourcentage de simplification

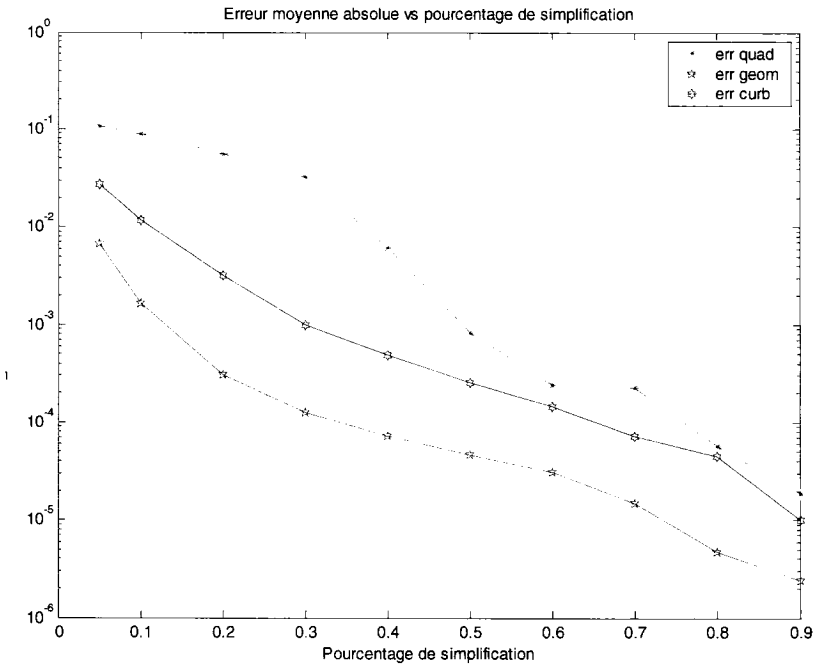



FIGURE 6.7 – Erreur moyenne versus pourcentage de simplification en échelle logarithmique

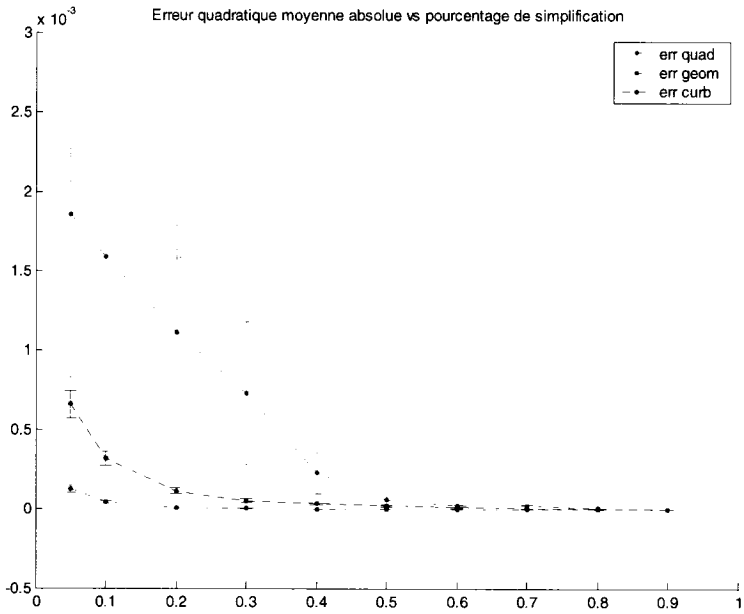



FIGURE 6.8 – Erreur quadratique moyenne versus pourcentage de simplification

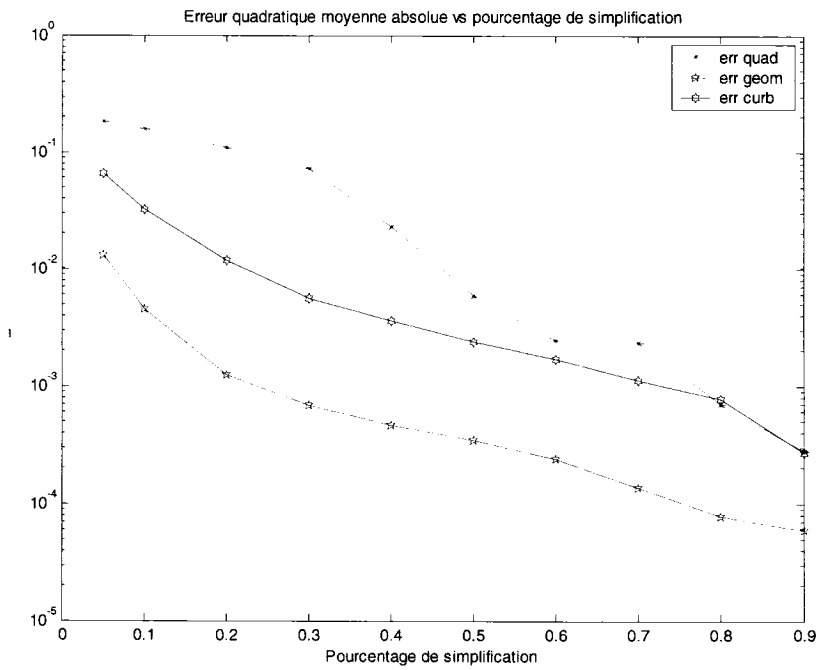



FIGURE 6.9 – Erreur quadratique moyenne versus pourcentage de simplification en échelle logarithmique

Les graphiques des trois types d'erreurs ont une forme presque identique et on peut noter que la différence entre la qualité d'approximation commence à se faire sentir surtout lorsque le pourcentage de réduction des triangles est élevé, soit à partir d'environ 50 %. Pour les trois types d'erreurs, l'algorithme de « Simplification des surfaces via l'erreur géométrique » performe mieux que l'algorithme de « Simplification des surfaces via l'erreur de courbure » et encore mieux que l'algorithme de « Simplification des surfaces via l'erreur quadratique ». On peut également remarquer que l'écart de qualité entre les trois algorithmes n'est pas linaire et ce, à partir d'environ de 50 % de réduction des triangles. Cet écart est plus apparent surtout au niveau des pourcentages élevés de réduction des triangles, soit à moins de 40 %. L'algorithme basé sur la simplification via l'erreur géométrique performe environ quatre fois mieux que l'algorithme basé sur la simplification via l'erreur de courbure, lequel performe lui-même environ quatre fois mieux que l'algorithme basé sur la simplification via l'erreur quadratique.

Les figures ci-dessous présentent des exemples de simplification d'une surface et ce pour les trois algorithmes :

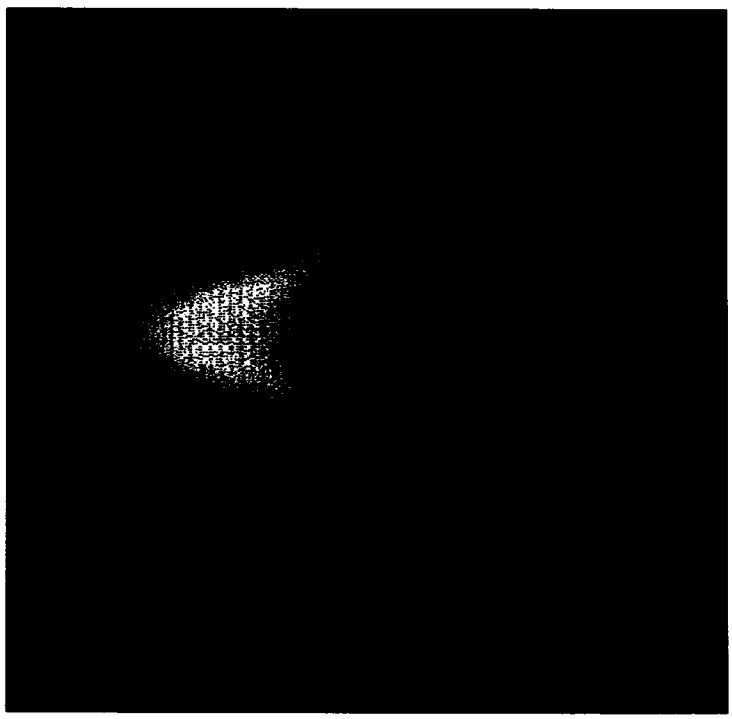
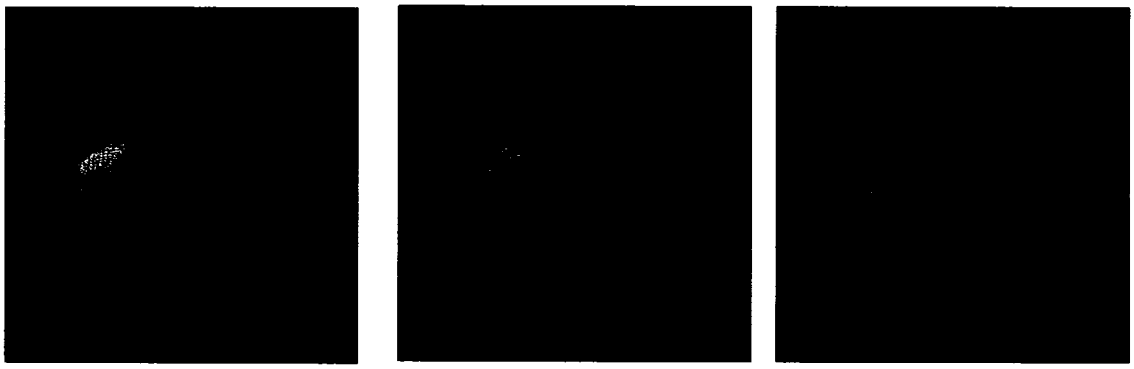


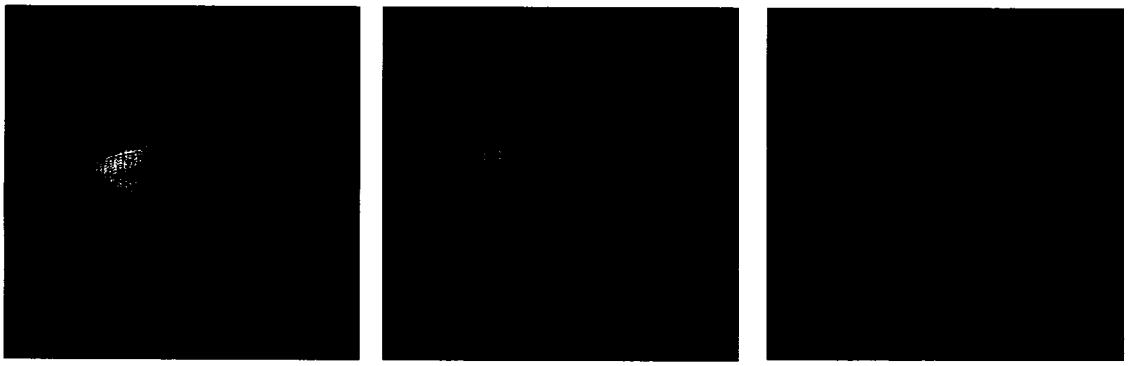
FIGURE 6.10 – Surface originale comportant 30 920 triangles

Ci-dessous, représentation de la surface ci-haut mais après simplification avec des pourcentages de réduction des triangles de 80 %, 90 % et 95 %, pour un nombre de triangles restant respectivement de 6 184, 3 092 et 1 545 :



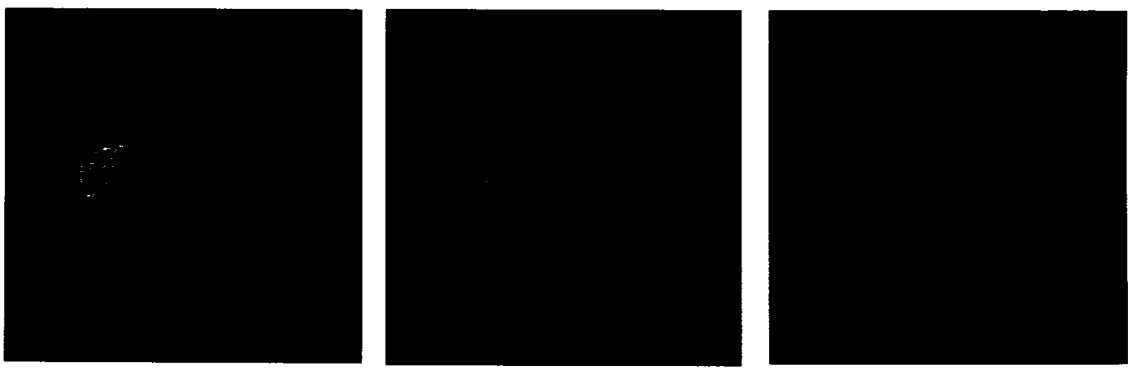
3 548 sommets 1 931 sommets 1 053 sommets

FIGURE 6.11 – Simplification de la surface originale via l'erreur quadratique



3 434 SOMMETS 1 762 SOMMETS 930 SOMMETS

FIGURE 6.12 – Simplification de la surface originale via l'erreur géométrique



3 130 SOMMETS 1 577 SOMMETS 797 SOMMETS

FIGURE 6.13 – Simplification de la surface originale via l'erreur de courbure

On peut remarquer sur les représentations ci-dessus que même pour des pourcentages de réduction maximal des triangles, soit de 95 %, la forme originale de la surface ainsi que ses frontières sont préservés. Néanmoins, l'arrangement des triangles diffèrent significativement dépendamment de l'algorithme. De plus, le nombre de sommets résultant de la simplification est différent pour les trois algorithmes.

### 6.1.3.2 Résultats de comparaison des surfaces de la deuxième catégorie

Le but de ces cas tests est de déterminer lequel des algorithmes performe le mieux en terme de préservation des frontières et des trous. C'est pour cette raison que l'analyse visuelle est aussi importante que l'analyse numérique pour ce type de surfaces. Nous débutons par l'analyse numérique et par la suite, présentons quelques exemples les plus flagrants d'erreurs produites par les différents algorithmes.

Les surfaces de la deuxième catégorie sont celles avec des trous, préalablement présentées à la section 5.1.2. Comme pour les surfaces sans trous, nous avons effectué la même démarche en vue de la récupération statistique des erreurs; la seule différence réside dans le fait qu'il n'y avait que huit (8) échantillons au lieu de 99. Comme pour les surfaces de la première catégorie, nous avons simplifié les surfaces avec trous selon dix (10) niveaux de simplification; chaque point sur la courbe représente la moyenne des huit surfaces selon le niveau de simplification. Les graphiques ci-dessous présentent les résultats des erreurs géométriques présentés comme ceux de la catégorie précédente.

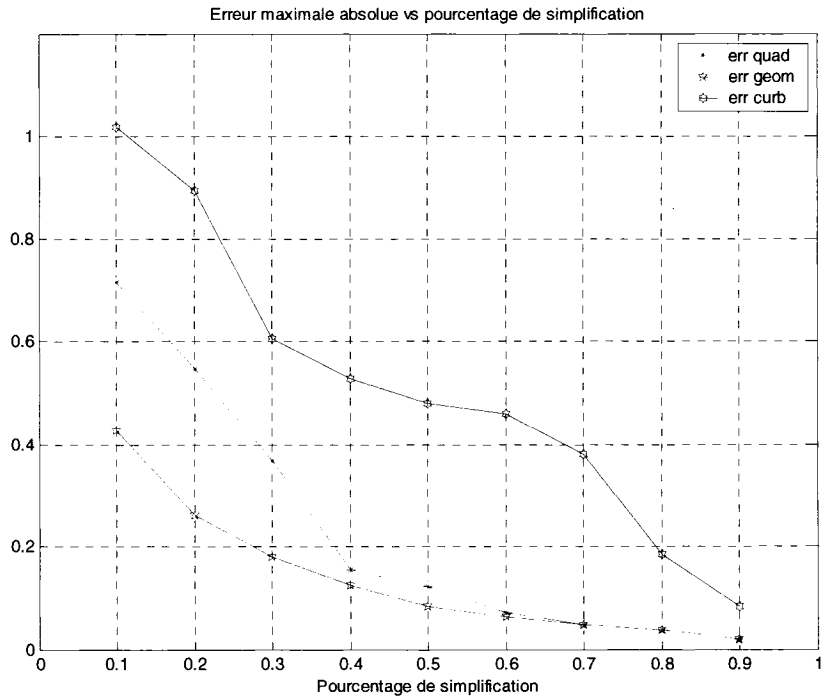



FIGURE 6.14 – Erreur maximale versus pourcentage de simplification

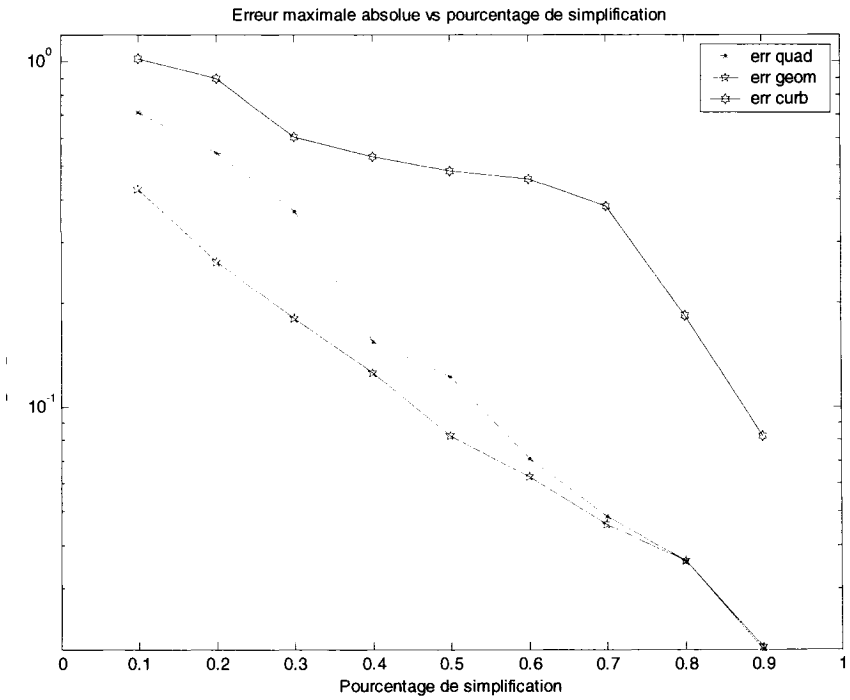



FIGURE 6.15 – Erreur maximale versus pourcentage de simplification en échelle logarithmique

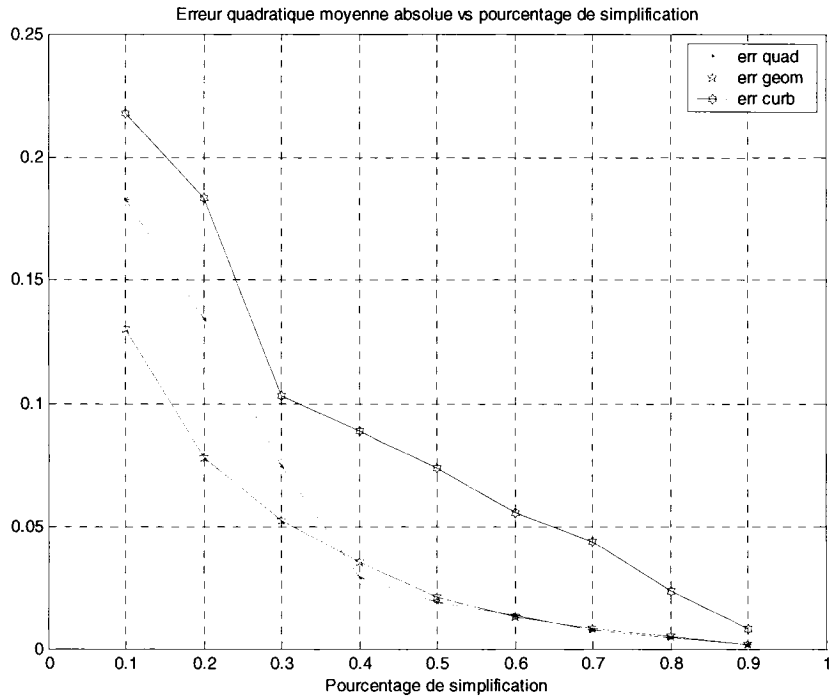



FIGURE 6.16 – Erreur moyenne versus pourcentage de simplification

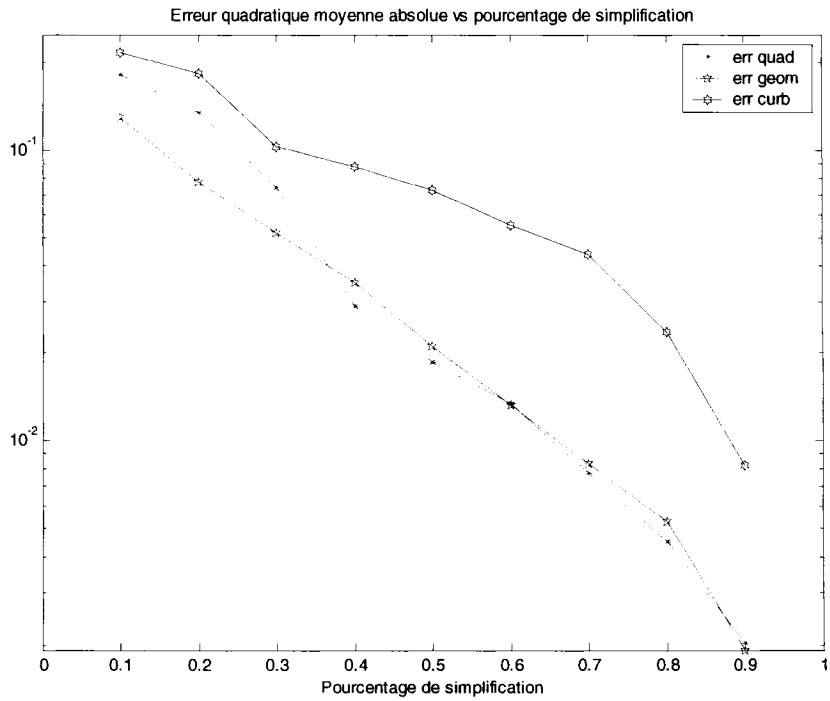



FIGURE 6.17 – Erreur moyenne versus pourcentage de simplification en échelle logarithmique

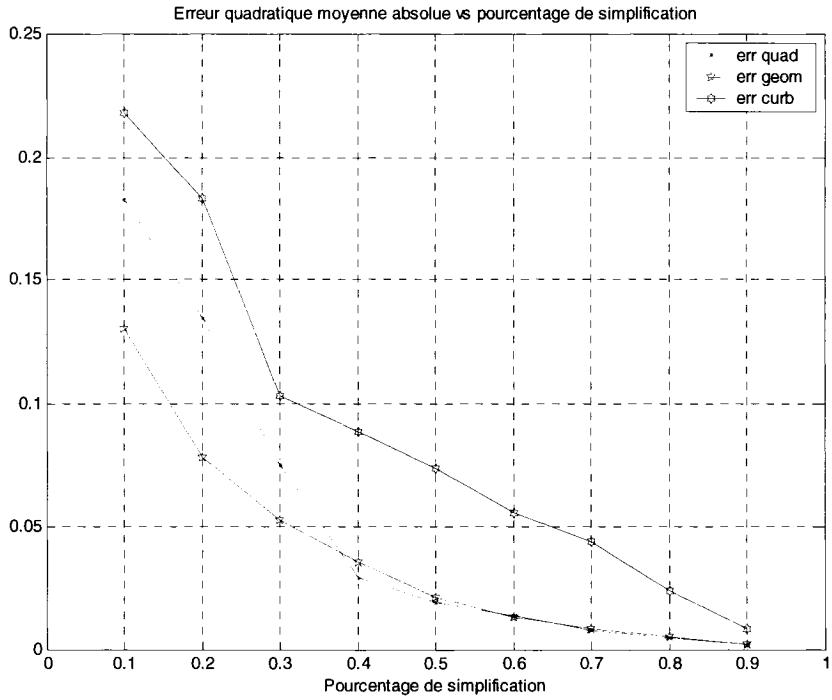



FIGURE 6.18 – Erreur quadratique moyenne versus pourcentage de simplification

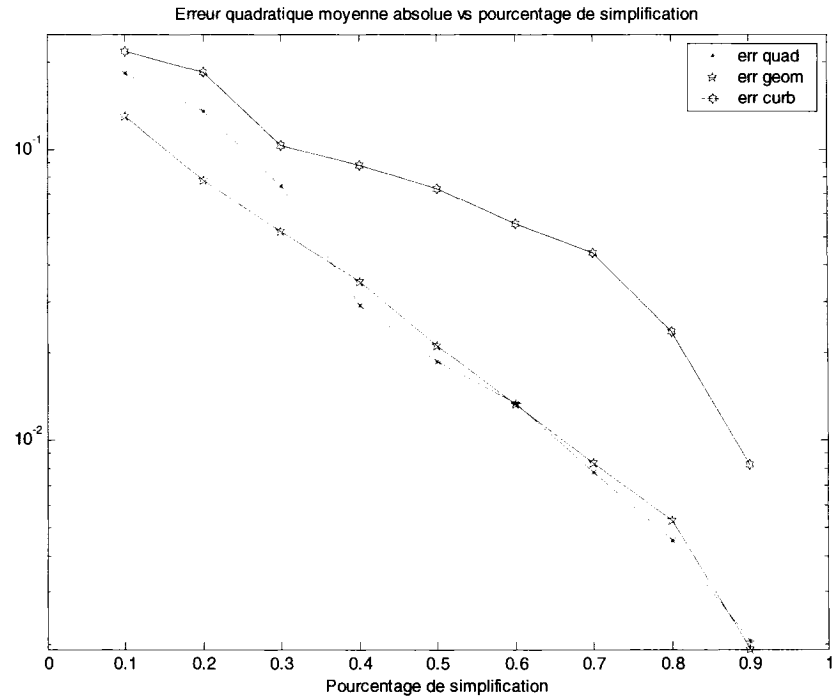



FIGURE 6.19 – Erreur quadratique moyenne versus pourcentage de simplification en échelle logarithmique

D'après ces résultats numériques, nous pouvons affirmer que la tendance démontrée à la section précédente signifie que l'algorithme basé sur l'évaluation de l'erreur géométrique performe mieux et se maintient pour cette catégorie aussi. Également, cette expérience confirme le fait que la différence en ce qui a trait à la qualité d'approximation est plus importante surtout pour un taux de simplification des triangles élevés, soit à partir de 60 % de triangles simplifiés. On peut aussi mentionner que l'algorithme « Simplification de surfaces via l'erreur géométrique » performe environ deux fois mieux que celui de « Simplification de surfaces via l'erreur quadratique », lequel à son tour performe deux fois mieux que celui de « Simplification de surfaces via l'erreur de courbure ».

Nous présentons ci-dessous un exemple de simplification d'une des surfaces de la deuxième catégorie , par les trois algorithmes.

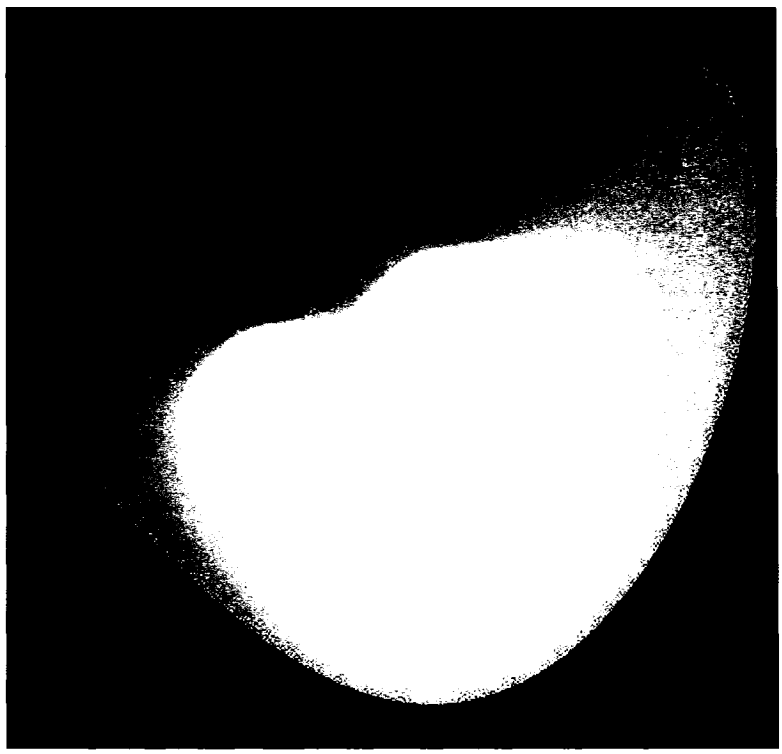
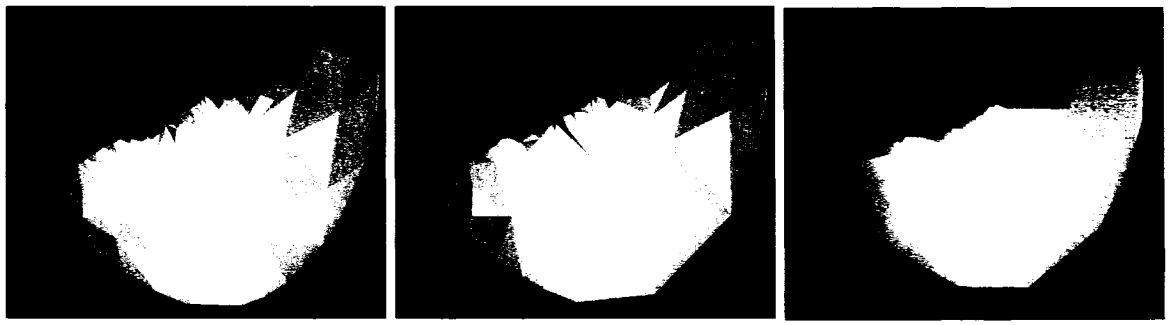


FIGURE 6.20 –Surface originale représentée par 2 068 triangles et 1 154 sommets, avec trou au milieu et frontières complexes

Les figures ci-dessous présentent la surface ci-haut mais après simplification avec des pourcentages de réduction des triangles de 80 %, 90 % et 95 %, pour un nombre de triangles restant respectivement de 413, 206 et 103 :

314 sommets 196 sommets 104 sommets

FIGURE 6.21 – Simplification de la surface originale via l'erreur quadratique



270 sommets 144 sommets 79 sommets

FIGURE 6.22 – Simplification de la surface originale via l'erreur géométrique

236 sommets 125 sommets 66 sommets

FIGURE 6.23 – Simplification de la surface originale via l'erreur de courbure.

Dans le présent exemple, il est possible de constater que l'algorithme basé sur l'erreur géométrique produit de meilleurs résultats que celui basé sur l'erreur quadratique, alors que ce dernier produit de meilleurs résultats que celui basé sur l'erreur de courbure et ce pour un taux de simplification élevé. Pour un taux de simplification moins élevé, tous les algorithmes produisent plus ou moins les mêmes résultats visuels

Ci-dessous un autre exemple de simplification d'une surface par les trois algorithmes :

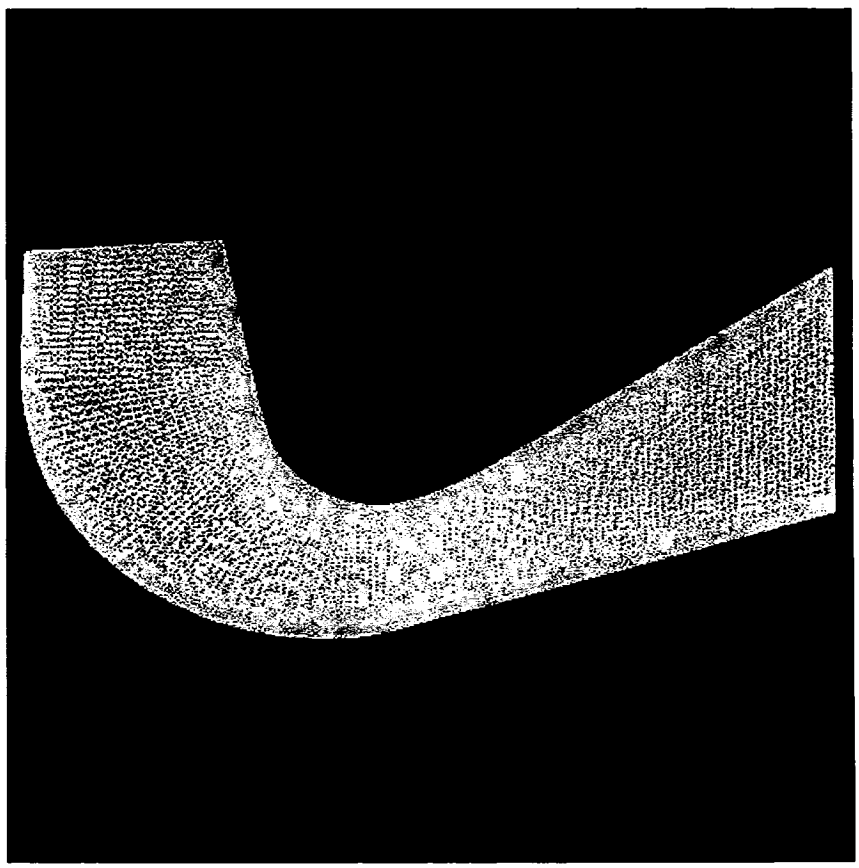
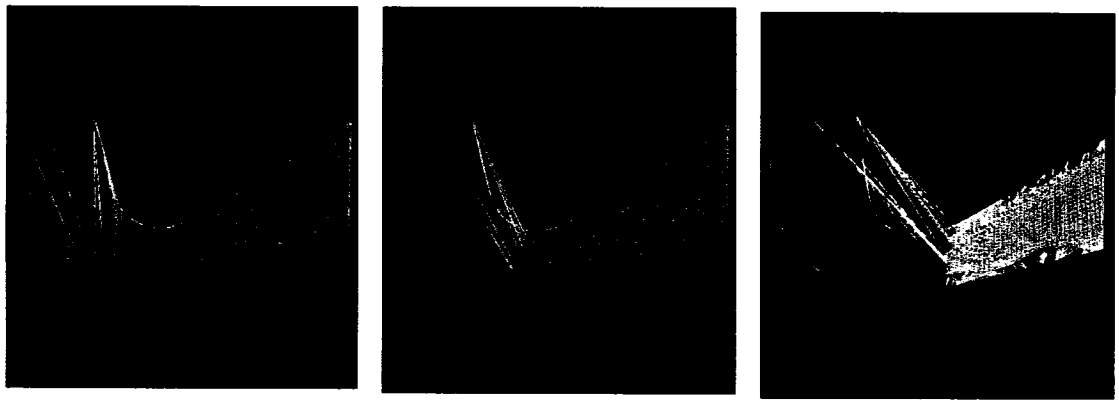


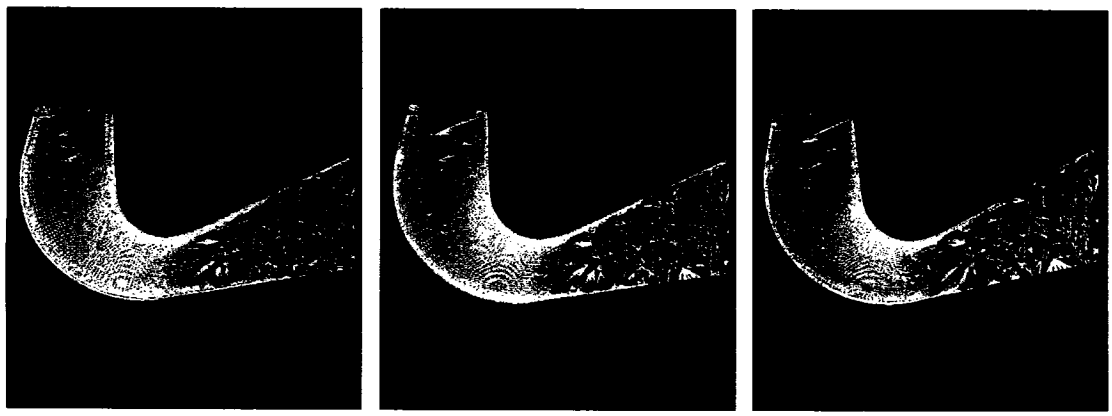
FIGURE 6.24 - Surface originale comportant 30 920 triangles

Ci-dessous représentation de la surface ci-haut mais après simplification avec des pourcentages de réduction des triangles de 50 %, 40 % et 30 %, pour un nombre de triangles restant respectivement de 15 459, 12 368 et 9 275.



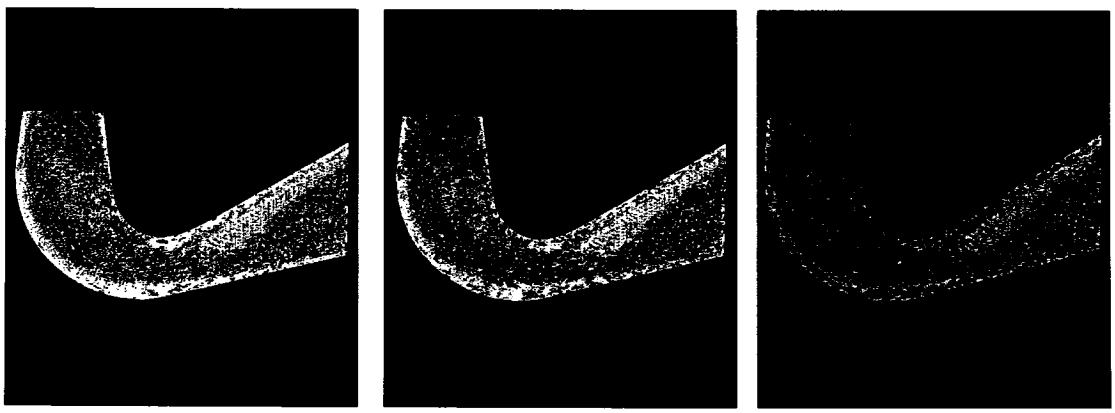
8 012 sommets 6 462 sommets 4 915 sommets

FIGURE 6.25 – Simplification de la surface originale via l'erreur quadratique



8 171 sommets 6 624 sommets 5 065 sommets

FIGURE 6.26 – Simplification de la surface originale via l'erreur géométrique



8 081 sommets 6 535 sommets 4 987 sommets

FIGURE 6.27– Simplification de la surface originale via l'erreur de courbure

Cet exemple a été intégré afin de démontrer une faille dans l'algorithme de « Simplification des surfaces via l'erreur quadratique ».

En effet, à la figure 6.25, comportant 8 012 sommets, on peut remarquer que la simplification a commencé par les triangles se trouvant à l'intérieur de la surface. Étant donné que ces triangles n'ont pas été identifiés comme triangles comportant des frontières, la simplification s'est poursuivie mais la forme des triangles résultant de cette simplification a dépassé les frontières extérieures de la surface (figure 6.25, comportant 6 462 sommets). Dans la poursuite de la simplification (figure 6.25 comportant 4 915 sommets), on peut constater que la forme originale de la surface est complètement détériorée comparativement à la surface originale. Pourtant la bande de triangles comportant les frontières extérieures originales a été préservée.

Il s'agit ici d'une grave erreur géométrique dans l'évaluation numérique. Visuellement le résultat diffère totalement de la surface originale.

Les deux autres simplifications représentées aux figures 6.26 et 6.27, notamment «Simplification de surfaces via l'erreur géométrique » et « Simplification de surfaces via l'erreur de courbure » ne présentent pas les mêmes problèmes.

#### 6.1.3.3 Résultats de comparaison des surfaces de la troisième catégorie

Le but de ce cas test était essentiellement de comparer les algorithmes selon leur capacité à conserver les détails. Nous avons donc procédé à la simplification des trois modèles dont il fut question à la section 5.1.3. Étant donné le nombre réduit de modèles, soit seulement trois, il s'avérait inutile de procéder à la comparaison numérique puisque les résultats n'étaient pas représentatifs; nous n'avons donc procédé qu'à la comparaison visuelle.

Pour ce faire nous avons procédé à la simplification des trois modèles selon dix niveaux de réduction, tel que décrit précédemment dans le présent rapport. Le dernier niveau de réduction pour les deux catégories de surfaces décrites précédemment s'arrêtait à un taux de simplification des triangles de 95 %.

Pour les modèles de la troisième catégorie, nous avons également procédé à la simplification des triangles jusqu'à l'obtention d'un taux de 95 % mais puisque nous n'avons pas noté de différence marquée entre les trois algorithmes, nous avons été dans l'obligation de simplifier encore davantage i.e. pour un taux de simplification à 97,5 % et à 98,75 %.

Les figures ci-dessous présentent les résultats obtenus pour ces deux derniers taux de simplification :

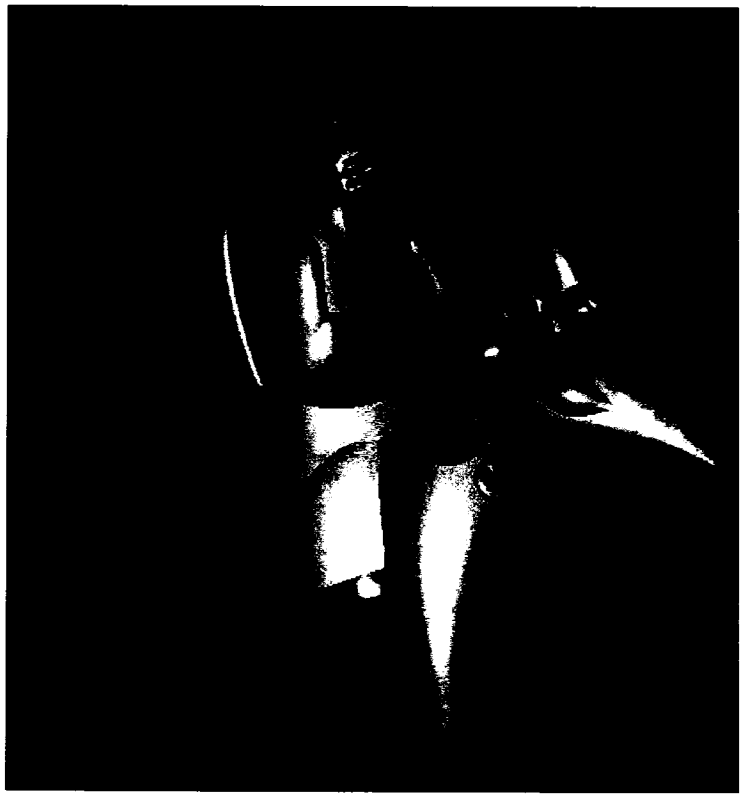
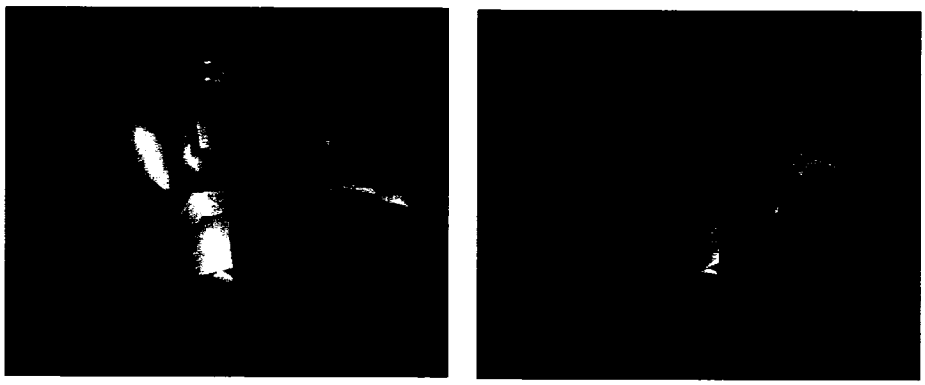


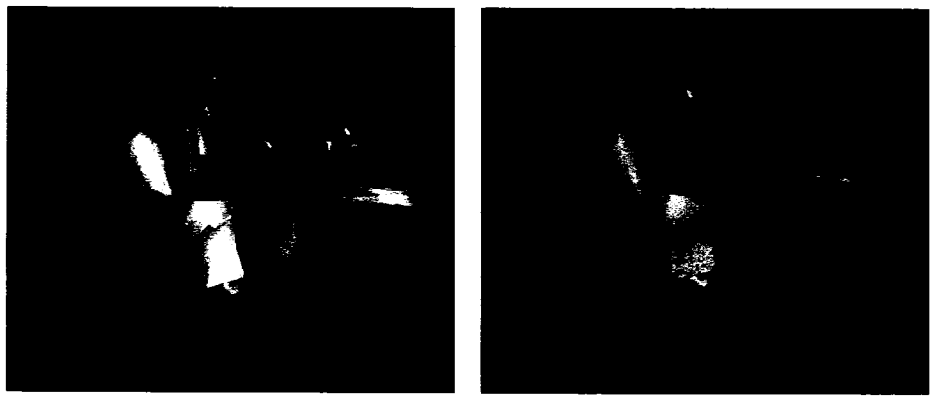
FIGURE 6.28 – Maillage d'un chevalier représenté par 93 286 triangles et 46 824 sommets.

Ci-dessous simplifications du maillage ci-haut pour un taux de réduction des triangles de 97,5 % et 98,75 % pour un nombre de triangles respectifs de 2 321 et de 1 150.



1 968 1 191

FIGURE 6.29 – Simplification du maillage « chevalier » via l'erreur quadratique



1 576 SOMMETS 858

FIGURE 6.30 – Simplification du maillage « chevalier » via l'erreur géométrique

1 463 771

FIGURE 6.31 – Simplification du maillage « chevalier » via l'erreur de courbure

Sur les figures représentées ci-haut, on pourra noter que pour une réduction des triangles de 98,75 % les meilleurs résultats ont été obtenus en ayant recours à la simplification selon l'erreur géométrique.

Ci-dessous exemples supplémentaires de comparaisons selon la préservation des détails :

FIGURE 6.32 - Maillage d'un lion représenté par 129 004 triangles et 64 634 sommets.

Ci-dessous simplifications du maillage ci-haut pour un taux de réduction des triangles de 97,5 % et 98,75 % pour un nombre de triangles respectifs de 3 221 et de 1 609.

2 682 SOMMETS 1 664 SOMMETS

FIGURE 6.33 - Simplification du maillage « lion » via l'erreur quadratique.

2 074 1 117 SOMMETS

FIGURE 6.34 - Simplification du maillage « lion » via l'erreur géométrique

2 016 1 143 SOMMETS

FIGURE 6.35 - Simplification du maillage « lion » via l'erreur de courbure

Une fois de plus dans ce dernier exemple, on peut constater que l'algorithme de simplification via l'erreur géométrique produit de meilleurs résultats que les autres. De plus, on peut remarquer que l'algorithme de simplification via l'erreur de courbure performe mieux que l'algorithme de simplification via l'erreur quadratique.

Ce dernier cas test effectué sur les surfaces de troisième catégorie nous amène à la conclusion que pour un pourcentage élevé de réduction de triangles l'algorithme de simplification via l'erreur géométrique préserve les détails initiaux du modèle de bien meilleure façon que les autres algorithmes.

## 6.2 Résumé des comparaisons des algorithmes et recommandations

Dans la section 6.1, nous avons présenté les résultats empiriques des comparaisons des trois algorithmes. La présente section sera dédiée aux résultats de comparaison les plus importants, ainsi qu'aux recommandations que nous sommes en mesure de faire quant à certaines implémentations possibles dans le logiciel Topovisu afin d'en améliorer les performances.

### 6.2.1 Efficacité selon le temps d'exécution et la consommation de mémoire

Le but premier d'un algorithme de simplification est de produire le plus rapidement possible une approximation qui ne différera que très peu du modèle original.

Concernant le temps d'exécution, l'algorithme « Simplification via l'erreur quadratique » de M. Garland est environ deux fois plus rapide que l'algorithme « Simplification via l'erreur géométrique » de M. Hussain et de dix à cent fois plus rapide que celui de « Simplification via l'erreur de courbure ». Ces résultats sont valides pour les surfaces d'écoulement et ont été confirmés par les résultats obtenus pour les surfaces avec trous et ce, même si le nombre de ces surfaces était restreint.

Quant à la consommation de mémoire, c'est l'algorithme « Simplification via l'erreur géométrique » de M. Hussain qui est meilleur que celui de la « Simplification via l'erreur quadratique » car ce dernier consomme 128 octets par sommet de plus, provoquant ainsi la pagination excessive de la mémoire virtuelle dans les cas de modèles composés d'un grand nombre de sommets.

### 6.2.2 Efficacité selon l'exactitude d'approximation

Tel que mentionné à la section précédente, l'efficacité d'un algorithme est jugé selon sa rapidité; cependant, la rapidité d'un algorithme peut être annulée par son inexactitude.

Quant à l'exactitude numérique, dans le cas des 99 surfaces d'écoulement l'algorithme « Simplification via l'erreur géométrique » performe environ quatre fois mieux que l'algorithme « Simplification via l'erreur de courbure » et environ dix fois mieux que l'algorithme « Simplification via l'erreur quadratique ». Dans les cas de 8 surfaces avec trous, ici encore l'algorithme « Simplification via l'erreur géométrique » a un rendement deux fois supérieur à celui « Simplification via l'erreur quadratique ». Ce dernier performe environ deux fois mieux que l'algorithme « Simplification via l'erreur de courbure ».

On peut remarquer que dans le cas des surfaces d'écoulement, l'algorithme basé sur l'évaluation d'erreur de courbure donne de meilleurs résultats que celui basé sur l'évaluation d'erreur quadratique. Dans le cas de surfaces avec trous, c'est l'algorithme basé sur l'évaluation d'erreur quadratique qui performe mieux que celui basé sur l'évaluation d'erreur de courbure.

Nous expliquons ces résultats par le fait que dans l'algorithme basé sur l'évaluation d'erreur quadratique la préservation des frontières est bien définie tandis que dans le cas de l'algorithme basé sur l'évaluation de courbure, la préservation des frontières n'est pas encore définie en terme d'évaluation d'erreur de courbure. Dans ce dernier algorithme, les opérateurs d'évaluation de courbure Gaussienne et de courbure moyenne ne sont applicables qu'aux sommets intérieurs de la surface et non aux sommets situés sur la frontière des surfaces.

Nous en concluons donc que l'algorithme basé sur l'évaluation d'erreur quadratique provoque moins d'erreurs pour les surfaces avec trous.

Concernant la capacité de préservation des détails, ici encore l'algorithme « Simplification via l'erreur géométrique » performe mieux et ce, surtout pour un taux de réduction de triangles très élevé, soit à près de 97 %. Le deuxième algorithme préservant mieux le niveau de détail initial est celui utilisant la « Simplification via l'erreur de courbure ».

### 6.2.3 Recommandations quant aux choix de l'algorithme à implanter

Le présent travail a été effectué dans le but d'en arriver à faciliter le choix d'un algorithme de simplification à implémenter dans Topovisu. À la lumière du présent travail, le choix de l'algorithme à implémenter dépendra de l'importance que l'on accorde respectivement à l'efficacité de l'algorithme, à sa capacité à traiter du problème de très grande taille et à la précision du résultat cherché.

En effet, si le premier critère est la rapidité, l'algorithme « Via l'erreur quadratique » devra être privilégié mais dans le cas où la consommation de mémoire réduite serait préconisée, le choix devrait se porter sur l'algorithme « Via l'erreur géométrique ».

Personnellement, comme nous avons contribué à la maintenance et au développement de Topovisu, nous sommes d'avis que l'algorithme via l'erreur géométrique répond mieux aux besoins du programme étant donné que le nombre de sommets est inconnu.

Quant à l'algorithme « Via l'erreur de courbure », considérant le type de surfaces utilisées par Topovisu, celui-ci n'est pas dans la course puisque son temps d'exécution est trop lent.

# CHAPITRE 7

# CONCLUSION

Dans le présent travail nous avons procédé à la comparaison de trois algorithmes de simplification les plus récents.

Le premier de ces trois algorithmes, la « Simplification des surfaces via l'erreur quadratique » (Garland, M., 1999), (Garland, M. et al., 1997) est actuellement un de ceux les plus reconnus dans le domaine de la simplification de surfaces pour sa rapidité et sa qualité d'approximation. Le deuxième, « Simplification des surfaces via l'erreur géométrique » (Hussain, M » et al., 2003), (Hussain, M. et al., 2001), est plus récent que le précédent et son principal avantage est sa consommation de mémoire réduite. Enfin, le troisième algorithme, « Simplification des surfaces via l'erreur de courbure » (Kim, S.-J. et al., 2002), (Kim, S.-J. et al., 2000), a été choisi principalement étant donné que sa métrique d'erreur est basée sur l'évaluation de la courbure et non sur l'évaluation de la distance.

## 7.1 Résumé des contributions

La comparaison des algorithmes de simplification n'est pas un sujet complètement nouveau. Ce qui différencie le présent travail des autres recherches est la méthodologie de comparaison des algorithmes. Nous incluons ci-dessous quelques points novateurs de notre méthodologie:

1. La comparaison numérique a été faite sur une base quantitative, i.e. que nous avons procédé à des tests empiriques sur plusieurs dizaines de surfaces ce qui, à notre connaissance, n'a jamais été fait auparavant et nous a permis d'analyser les résultats de façon statistique. Une quantité minime de résultats n'est pas très représentative statistiquement alors qu'une plus grande quantité de tests nous a permis de tirer des conclusions avec une assurance accrue dans le cas des 99 surfaces du domaine hydroélectrique.

2. L'implémentation des trois algorithmes s'est faite sur une base unique, i.e. que les trois algorithmes ont été implémentés sur la même base, ce qui n'a également jamais été fait auparavant. Le fait de travailler de cette façon a rendu les résultats de comparaison plus crédibles puisqu'effectués sur la même machine et avec les mêmes ressources.

3. Pour les cas test, nous avons utilisé trois types de surfaces :
   - Des surfaces issues du domaine industriel, dont 99 sans trous, ont principalement été utilisées lors de comparaisons quantitatives.
   - Des surfaces de forme complexe avec trous ont également été utilisées, soit au nombre de 8 afin de comparer la capacité de l'algorithme à conserver la topologie avec les trous.
   - Des modèles complexes issus du format .nurbs, au nombre de 3, ont été utilisés afin de comparer la préservation du niveau de détail initial.

4. Nous avons défini en détail l'algorithme de « Simplification des surfaces via l'erreur de courbure discrète » afin de l'implémenter. Nous l'avons par la suite amélioré en intégrant l'opérateur de courbure moyenne dont la définition se trouve dans l'article de M. Meyer et al., 2002.

## 7.2 Résumé des comparaisons des trois algorithmes

Nous pouvons affirmer que l'algorithme « Simplification des surfaces via l'erreur quadratique » est plus rapide que les deux autres. En terme de temps d'exécution, celui-ci est environ deux fois plus rapide que l'algorithme de « Simplification des surfaces via l'erreur géométrique » et une dizaine de fois plus rapide que l'algorithme de « Simplification des surfaces via l'erreur de courbure ».

Pour ce qui est de la consommation de mémoire, nous avons éprouvé des problèmes de pagination virtuelle excessive avec l'algorithme de « Simplification des surfaces via l'erreur quadratique », ce que nous qualifions de faille importante de cet algorithme.

Quant à la qualité des approximations, l'algorithme de « Simplification des surfaces via l'erreur géométrique » est plus précis que celui « via l'erreur quadratique », surtout pour un niveau élevé de réduction de triangles pour toutes les catégories de surfaces que nous avons testées.

En ce qui concerne la préservation des frontières et des trous, même si la taille d'échantillons dans ce cas a été réduite, ici encore l'algorithme de « Simplification des surfaces via l'erreur géométrique » préserve mieux les frontières et les trous que celui « via l'erreur quadratique », ce qui corrobore les dires de l'auteur de l'algorithme dans son article (Hussain, M., 2004).

Quant à la préservation des détails initiaux d'un modèle, une fois de plus nous en venons à la conclusion que l'algorithme de « Simplification des surfaces via l'erreur géométrique » donne les meilleurs résultats et ce, toujours avec un nombre réduit de modèles, toujours en conformité avec les dires de son auteur, M. Hussain (2004).

Cet algorithme pourrait donc être très utile dans le cas d'optimisation de jeux par exemple, car les jeux utilisent habituellement beaucoup de modèles représentés par un grand nombre de triangles et les ressources de mémoire sur les consoles de jeux sont limitées.

Quant au troisième algorithme, soit « via l'erreur de courbure », il se classe parmi les moins performants pour le temps d'exécution mais également pour la qualité d'approximation. Le principe de cet algorithme est quand même prometteur mais n'est pas encore assez bien élaboré, surtout quant à la simplification des sommets se trouvant aux frontières de la surface. Il peut cependant être très utile pour certains types de surfaces dont la courbure locale est importante.

En résumé, nous pouvons conclure que l'algorithme de « Simplification des surfaces via l'erreur géométrique » est une excellente alternative à l'algorithme « via l'erreur quadratique » surtout dans les cas de simplification de modèles représentés par un très grand nombre de triangles.

## 7.3 Suggestions pour travaux futurs

À la lumière des résultats que nous avons obtenus, les travaux futurs pourraient se diriger dans diverses directions.

Premièrement, tel que mentionné, l'implémentation de l'algorithme de simplification dans le logiciel Topovisu pourrait être faite afin d'améliorer la performance d'affichage des surfaces NURBS.

Également, l'algorithme de simplification de surfaces via l'erreur de courbure n'est pas complètement défini pour les sommets se trouvant sur la frontière des surfaces.

Dans la version actuelle, nous traitons les sommets frontaliers selon l'algorithme de M. Hussain. Dans les travaux futurs, il serait intéressant de définir les opérateurs de courbure Gaussienne et moyenne pour les sommets frontaliers, ce qui permettrait d'évaluer la courbure discrète des sommets frontaliers et finaliser l'algorithme quant aux parties frontalières.

Finalement, tel que nous le faisions remarquer à la section 6.1.3.2, une anomalie dans l'algorithme de M. Garland a été détectée quant à la préservation des frontières. Un travail dans le but de corriger cette anomalie pourrait également être effectué ultérieurement.

# RÉFÉRENCES

ASPERT, N., SANTO-CRUZ, D.et EBRAHIMI, T. (2002), MESH: Measuring Errors between Surfaces using the Hausdorff distance, IEEE International conference in Multimedia and Expo (ICME), vol.1, pages 705-708

CAMARERO, R. (2003), Notes de cours: Génération de maillage

CIGNONI, P. MONTANI, P., SCOPIGNO, R. (1997), A comparison of mesh simplification algorithms, Computer and Graphics

CIGNONI, P., ROCCHINI, C. et SCOPIGNO, R. (1998), Metro : Measuring error on simplified surfaces, Computer graphics Forum, Vol.17, N°2, pages 165-174

COURSE NOTES (2000), Subdivision for modeling and animation. SIGGRAPH, pages 47-63, 183-194

DOCARMO, M.-P. (1976), Differential Geometry of Curves and Surfaces. Englewood Cliffs, New Jersey

GARLAND, M. (1999), Quadric-Based Polygonal Surface Simplification, PhD thesis, School of Computer Science, Carnegie Mellon University

GARLAND, M. et HECKBERT, P.-S. (1997), Surface Simplification Using Quadric Error Metrics, SIGGRAPH.

GIENG, T.S., HAMANN, B., JOY, K.I., SCHUSSMAN, G.L. et TROTTS, I.J. (1997), Smooth hierarchical surface triangulations, Proc IEEE Visualization.

GUIBAULT, F. (2005), Notes de cours : Conception géométrique assistée par ordinateur et visualisation

HOPPE, H. (1996), Progressive meshes, SIGGRAPH '96

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J. et STUETZLE, W. (1993), Mesh optimization, SIGGRAPH, pages 19-26

HUSSAIN, M., OKADA, Y. et NIIJIMA, K. (2001), Fast Simple and Memory Efficient Mesh Simplification, International Conference of Computer Graphics and Imaging (CGIM), pages 72-77

HUSSAIN, M., OKADA, Y., NIIJIMA, K. (2003), Fast, simple, feature preserving and memory efficient simplification of triangle meshes, International Journal of Image and Graphics, Vol. 3, N°4, pages 653-670

HUSSAIN, M., OKADA, Y. et NIIJIMA, K. (2004), Efficient and Feature-Preserving Triangular Mesh Decimation, Journal of WSCG, 2004, Vol.12, N°1-3

KIM, S.-J., JEONG, W.-K., KIMIN, C.-H., LOD generation with discrete curvature error metric. Proceedings of 2nd Korea Israel Bi-National Conference on Geometrical Modeling and Computer Graphics in the WWW Era, pp.97-104

KIM, S.-J., KIM,S.-K., et KIM, C.-H. (2002), Discrete Differential Error Metric for Surface Simplification, In Proceedings of Pacific Graphics, pp. 276-283, Beijing

LINDSTROM, P. et TURK, G. (1999), Evaluation of Memoryless Simplification, IEEE Transactions on visualization and computer graphics, vol. 5, N°2

LORENSEN, W.E., CLINE, H.E. (1997), Marching Cubes: A high resolution 3D surface construction algorithm. SIGGRAPH, vol. 21, n°4.

MEYER, M.., DESBRUN, M., SCHRODER, P. et BARR, A.-H. (2002), Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. VisMath, Berlin.

ROY, M. (2001), Qualité d'un modèle 3D simplifié, Mémoire de DEA, instrumentation et informatique de l'image, Université de Bourgogne

SCHROEDER, W.-J., ZARGE, J.-A. et LORENSEN, W.-E. (1992), Decimation of triangle meshes, SIGGRAPH, Vol. 26, N°2

SOUTHERN, R., MARAIS, P. et BLAKE, E. (2001), Generic Memoryless Polygonal Simplification, SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques, ACM Press, New York

# ANNEXES A

# NOTES D'IMPLÉMENTATION

## A.1 Description du programme utilisé lors de la récupération des maillages à partir de l'interface NURBS d'OpenGL.

Nous avons mis en place une structure de données permettant d'acquérir les données des sommets et les directives de restitution. Pour ce faire, nous avons créé quatre classes telles « vertex », « triangle », « polygon » et « polydata ».

La classe « vertex » représente l'information géométrique sur la position dans l'espace 3D d'un sommet tandis que la classe « polygon » sert à conserver l'information sur les types de polygones et emmagasine alors l'information quant aux sommets composant le polygone. La classe « polydata » quant à elle, par l'intermédiaire d'agrégations, contient l'ensemble des polygones nécessaires lors du traçage de la surface. Ci-dessous figure du diagramme de classes représentant l'architecture du programme :

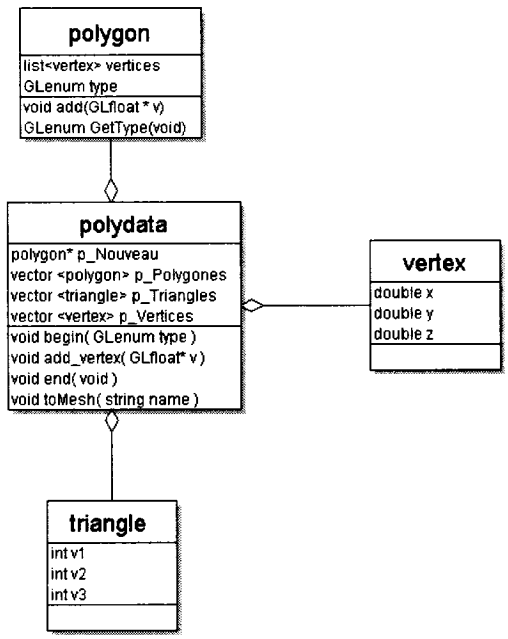



FIGURE A1 – Diagramme de classes de récupération de maillages

La classe « polydata » possède ainsi les méthodes nécessaires à l'extraction des données par l'intermédiaire de l'interface de rappel des NURBS, soit « begin », « add_vertex » et « end ». Lorsque l'interface de rappel des NURBS appelle la méthode « begin », cette dernière crée un nouveau polygone dont le type correspond à celui passé à l'interface. Après quoi l'interface appelle la méthode « add_vertex ». Celle-ci crée un nouveau sommet et emmagasine ce sommet dans le polygone nouvellement créé lors de l'appel de la fonction « begin ».

Cette séquence se répète jusqu'à ce que l'interface fasse appel à la méthode « end » dans laquelle le polygone est finalisé et placé dans la structure de données. Durant ce processus d'acquisition des sommets, les sommets redondants sont rejetés et les types de primitives géométriques associés aux polygones sont enregistrés.

Lorsque le rendu graphique d'une surface est complété, nous sommes alors à ce moment en possession de toute l'information relative à la position et à l'arrangement des sommets. L'arrangement des sommets dans le maillage triangulaire final est effectué par la fonction « toMesh ». Lors de la restitution d'une surface NURBS par OpenGL, les quatre types de primitives sont possibles :

1. GL_TRIANGLES
2. GL_TRIANGLE_FAN
3. GL_TRIANGLE_STRIP
4. GL_QUAD_STRIP

La fonction « toMesh », dépendamment du type de polygone, créera un ou plusieurs objets du type « triangle ». Les triangles créés contiendront les indices des sommets dans l'ensemble ordonné des sommets du maillage.

Nous pouvons donc procéder à la sauvegarde d'un maillage triangulaire dans un fichier ascii.

* *Note* : Pour récupérer les maillages des modèles complexes mentionnés à la section 5.1.3 (.nurbs), nous avons adopté exactement la même démarche que celle décrite ci-dessus. Dans le présent cas, nous avons cependant travaillé avec un logiciel de Kenny Hoffman, soit OpenGL NurbsViewer, destiné à la visualisation des modèles décrits en formats .nurbs et .bez, auquel nous avons apporté les mêmes modifications que pour Topovisu.

## A.2 Implémentation des trois algorithmes de simplification sur une base unique

### Design de classes

La structure de données de notre programme de simplification de maillages triangulaires contient cinq (5) classes, telles « Vec3 », « vertex », « triangle », « Mesh » et « Pmesh ». Pour parvenir à simplifier les maillages triangulaires, nous avons séparé les méthodes de simplification (les algorithmes) et les données sur lesquelles nos algorithmes ont été appliquées.

Notre programme contient les deux classes principales :

1. Mesh : Représente les données
2. Pmesh : Représente les méthodes de simplification

La classe Mesh est une structure servant à la description d'une surface polygonale. La classe Pmesh quant à elle sert à appliquer les algorithmes de simplification sur une surface donnée.

Sur la page suivante, vous trouverez le diagramme de classes.

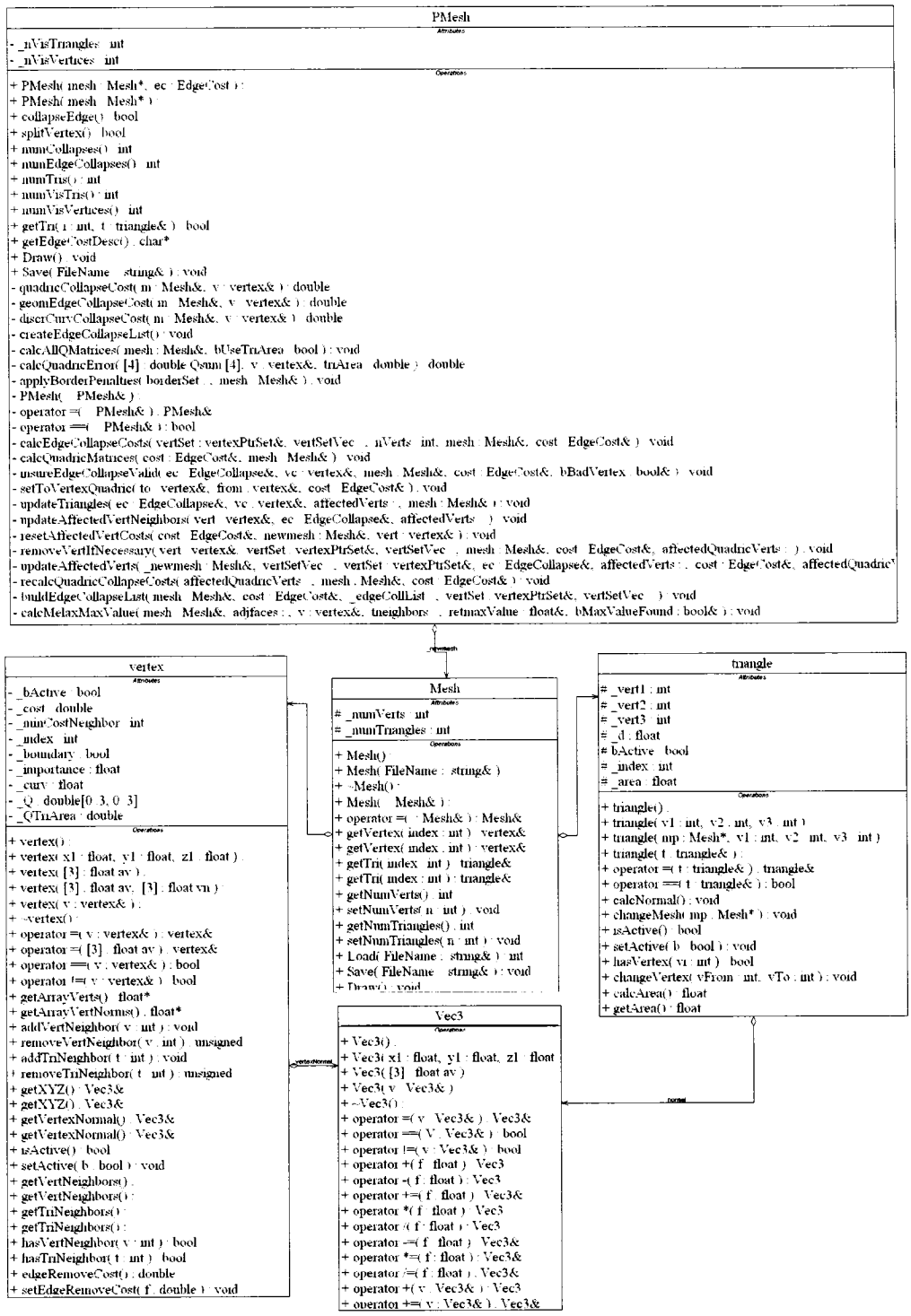



FIGURE A2 – Diagramme de classes du programme de simplification de maillages

## Description de classes

### Classe « Vertex » :

En ce qui concerne la classe « sommets », celle-ci englobe l'information géométrique quant au placement du sommet et sa normale, en plus de l'information en rapport avec les autres sommets et triangles qui lui sont connexes.

Cette classe possède de plus l'information à savoir si les sommets sont actifs ou pas (allumés ou non) mais également l'information quant au sommet voisin le moins coûteux à supprimer parmi tous ceux qui l'entourent. Fait à noter, elle possède également l'information en ce qui concerne son propre coût de suppression.
Finalement, l'instance de cette classe possède l'information et les méthodes servant à calculer l'importance visuelle de ce sommet (pour la méthode de l'erreur géométrique) et l'erreur quadratique (pour la méthode de l'erreur quadratique).

### Classe « Triangle » :

Quant à la classe « triangle », celle-ci possède l'information sur un triangle composant le maillage. Cette information comporte les indices des sommets composant le triangle, sa superficie, sa normale, de même que l'information à savoir si le triangle est allumé ou non et le pointeur sur le maillage dont ce triangle fait partie et son index dans la liste des triangles du maillage.

### Classe « Mesh » :

La classe « Mesh » possède les structures de données servant à conserver les informations quant aux positions des « sommets » du maillage et également l'information quant à la connectivité des triangles. Les structures de données représentent simplement les vecteurs de types « sommet » et « triangle » décrits plus haut. Évidemment, celle classe comprend la méthode servant à acquérir les données à partir d'un fichier texte, ainsi que les méthodes permettant l'accès aux sommets et triangles du maillage.

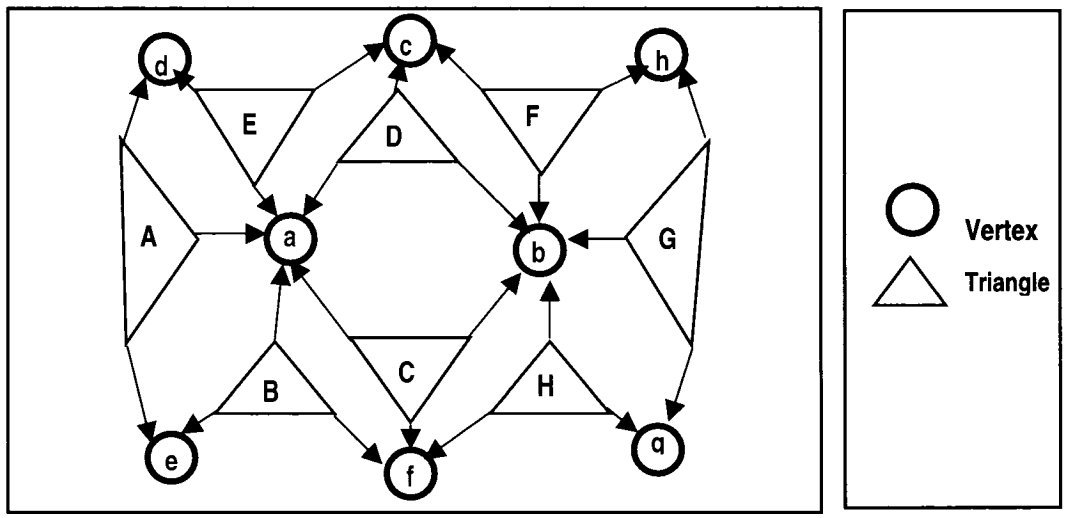



FIGURE A3 - Relation entre les différents objets de structure de données

Sur la figure ci-dessus, sommet « a » possède la liste des sommets voisins qui contiennent les sommets « e », « f », « b », « c » et « d ». Le sommet « a » possède également la liste des triangles dont il est le point commun. Cette liste contient les triangles « A », « B », « C », « D » et « E ».

**Classe « PMesh » :**

Cette classe joue un rôle primordial dans notre programme car c'est elle qui possède toutes les méthodes permettant les calculs et la simplification du maillage. En plus, cette classe possède deux objets de type « Mesh » ; ces objets servent à conserver l'information en ce qui concerne le maillage du contrôle, soit celui qui ne varie jamais, ainsi que l'information sur le maillage progressif, soit celui non statique en cours d'exécution du programme.

**Description des méthodes de la classe « PMesh »**

La construction de la classe « PMesh » débute en faisant appel à la méthode « createEdgeCollapseList() » dans laquelle, dépendamment de l'algorithme de simplification, le calcul des coûts de suppression pour chacun des sommets du maillage s'effectue en recourant à la méthode « calcEdgeCollapseCosts ». La structure de données contenant les pointeurs sur les sommets est alors initialisée selon les coûts de suppression des sommets, en ordre croissant.

Par la suite, il est fait appel à la méthode « buildEdgeCollapseList() ». Cette méthode récupère la structure initialisée des pointeurs sur les sommets et pour chacun des sommets, par l'intermédiaire de la méthode « insureEdgeCollapseValid » construit un objet de type « EdgeCollapse » qui comprend les indices des sommets à fusionner.

Cette dernière méthode vérifie par la suite la pertinence de la suppression de l'arête venant d'être construit. Par exemple, on pourrait imposer des conditions quant aux arêtes frontalières dont la suppression n'est pas requise. Dans un second temps, la mise à jour de l'ensemble des triangles et des coûts de suppression des sommets affectés par la fusion est effectuée par l'intermédiaire des méthodes « updateTriangles » et « updateAffectedVertex ». Cette dernière méthode effectue le recalcule des coûts de suppression des sommets affectés et procède à la mise à jour de la structure globale des pointeurs sur les sommets, triés selon leurs coûts de suppression. Ce cycle se répète jusqu'à ce que tous les sommets aient été vérifiés aux fins de suppression. Après l'itération décrite ci-dessus, un objet de type « EdgeCollapse » est initialisé et emmagasiné dans la liste afin de permettre l'identification de la prochaine arête à supprimer lors de décimation du maillage.

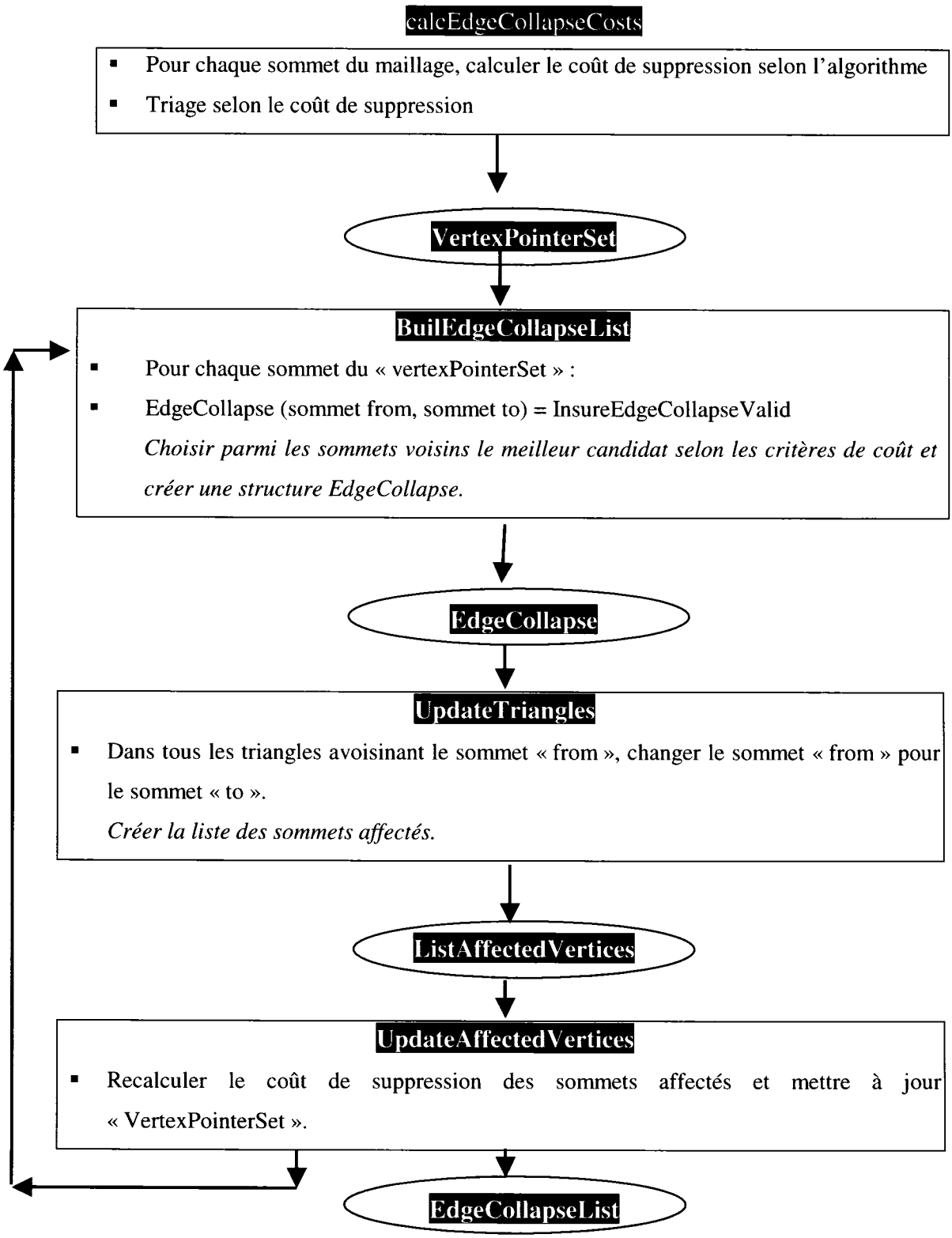



FIGURE A4- Représentation graphique du pseudo-code de l'algorithme

# ANNEXES B

# RÉSULTATS NUMÉRIQUES

## B.1 Listing du programme Matlab, contenant les données des comparaisons numériques, qui affiche les graphiques des figures 6.1 et 6.2

```
clear all;
%90% de confiance
student_table =[6.313752 2.919986 2.353363  2.131847 2.015048 1.943180 1.894579...
1.859548 1.833113 1.812461 1.795885 1.29007];

x=[550  2000  2100  3800  3900  8539 16189];%valeurs # de sommets
%valeurs de temps de l'algo basé sur l'erreur quadratique
%tq1 contient les valeurs cumulées pour les surfaces avec 550 nb de sommets
%tq2 contient les valeurs cumulées pour les surfaces avec 2000 nb de sommets
%tq3 -> 2100 , etc
tq1=[0.625 0.828];
tq2=[1.343 1.360 1.312 1.375 1.718 1.969];
tq3=[1.297 1.281 1.343 1.453 2.453 2.750];
tq4=[4.984 2.703 7.484 2.719 2.656 2.641...
     2.703 4.235 5.547 6.203 3.172 5.265];
tq5=[2.594 3.000 2.907 4.234 6.016 6.140...
     6.797 8.984];
tq6=[6.469 6.329 7.141 6.782 6.547 6.516...
     6.531 6.563 8.266 8.703 9.079 9.796];
tq7=[16.937 13.687 15.062 13.032 14.547...
    13.203 13.672 12.782 15.094 15.765...
    17.625 13.859 14.453 13.703 13.219...
    13.938 13.641 14.688 14.532 13.594...
    15.453 13.640 22.875 23.938 24.640...
    25.281 24.828 21.047 24.719 21.938...
    13.672 13.360 14.625 15.547 18.093...
    19.000 22.203 26.625 27.047 27.938...
    28.437 32.515 33.812 34.359 34.485...
    30.000 30.328 31.297 31.640 36.141...
    35.828 38.078 37.703];

%calcul des moyennes de
%        550     2000     2100     3800     3900     8539     16189
MoyTempQadr=[mean(tq1) mean(tq2) mean(tq3) mean(tq4) mean(tq5) mean(tq6) mean(tq7)];
```

```
% calcul de l'intervalle de confiance selon la distribution de Student

ErrTempQadr=[student_table(length(tq1)-1)*std(tq1)/sqrt(length(tq1))...
        student_table(length(tq2)-1)*std(tq2)/sqrt(length(tq2))...
        student_table(length(tq3)-1)*std(tq3)/sqrt(length(tq3))...
        student_table(length(tq4)-1)*std(tq4)/sqrt(length(tq4))...
        student_table(length(tq5)-1)*std(tq5)/sqrt(length(tq5))...
        student_table(length(tq6)-1)*std(tq6)/sqrt(length(tq6))...
        student_table(12)*std(tq7)/sqrt(length(tq7))];

%valeurs de temps de l'algo basé sur l'erreur géométrique
tg1=[1.703 1.938];
tg2=[4.985 5.094 5.297 4.453 3.750 3.734];
tg3=[6.000 5.047 5.422 4.390 4.453 4.500];
tg4=[7.219 7.421 7.313 7.344 7.625 7.204...
     8.406 7.234 7.282 7.234 8.766 8.532];
tg5=[8.687 8.687 8.297 10.078 8.828 8.704...
   8.485 8.469];
tg6=[31.938 35.969 38.313 37.625 31.968 29.688...
   31.156 28.657 25.016 25.984 25.609 26.625];
tg7=[50.469 47.250 53.312 48.781 43.187 51.500...
   42.406 51.375 41.578 51.672 41.672 42.891...
   46.312 44.359 51.125 48.156 43.985 49.125...
   51.921 49.250 54.906 51.406 46.188 54.735...
   46.813 52.938 53.094 47.829 51.766 48.563...
   43.719 52.672 44.625 55.844 41.313 41.000...
   41.187 67.969 44.797 65.328 58.079 44.938...
   47.859 44.547 47.625 49.063 50.875 48.328...
   52.796 45.125 48.109 44.781 47.812];

%calcul des moyennes de
%        550     2000    2100    3800    3900    8539    16189
MoyTempGeom=[mean(tg1) mean(tg2) mean(tg3) mean(tg4) mean(tg5) mean(tg6) mean(tg7)];


% calcul de l'intervalle de confiance selon la distribution de Student
ErrTempGeom=[student_table(length(tg1)-1)*std(tg1)/sqrt(length(tg1))...
        student_table(length(tg2)-1)*std(tg2)/sqrt(length(tg2))...
        student_table(length(tg3)-1)*std(tg3)/sqrt(length(tg3))...
        student_table(length(tg4)-1)*std(tg4)/sqrt(length(tg4))...
        student_table(length(tg5)-1)*std(tg5)/sqrt(length(tg5))...
        student_table(length(tg6)-1)*std(tg6)/sqrt(length(tg6))...
        student_table(12)*std(tg7)/sqrt(length(tg7))];

%valeurs de temps de l'algo basé sur l'erreur de courbure
tc1=[1.734 21.297];
tc2=[26.547 31.250 28.688 35.579 30.141 64.812];
tc3=[20.000 20.109 20.859 20.266 50.547 21.578];
```

```
tc4=[76.063 85.500 73.172 128.172 71.375 71.031...
     72.141 70.469 116.187 88.843 98.719 62.344];
tc5=[45.016 44.562 38.297 38.766 41.813 40.703...
     46.235 39.437];
tc6=[122.812 129.953 121.500 116.797 127.609 144.922...
     128.782 144.562 108.797 226.359 224.782 101.890];

tc7=[403.453 371.656 403.750 352.375 244.734 248.703...
   245.579 244.360 246.125 250.062 247.265 246.360...
   410.282 376.062 401.531 426.063 308.219 298.078...
   310.921 312.375 270.093 238.828 267.656 232.422...
   224.609 207.907 249.859 222.328 242.719 188.219...
   240.640 278.016 354.031 280.625 253.672 253.532...
   265.625 264.235 536.579 271.454 585.516 511.547...
   369.000 333.985 392.016 390.609 210.860 312.516...
   2039.14 369.641 325.843 322.125 296.625];

%calcul des moyennes
%        550     2000     2100     3800     3900     8539     16189
MoyTempCour=[mean(tc1) mean(tc2) mean(tc3) mean(tc4) mean(tc5) mean(tc6) mean(tc7)];

% calcul de l'intervalle de confiance selon la distribution de Student
ErrTempCour=[student_table(length(tc1)-1)*std(tc1)/sqrt(length(tc1))...
        student_table(length(tc2)-1)*std(tc2)/sqrt(length(tc2))...
        student_table(length(tc3)-1)*std(tc3)/sqrt(length(tc3))...
        student_table(length(tc4)-1)*std(tc4)/sqrt(length(tc4))...
        student_table(length(tc5)-1)*std(tc5)/sqrt(length(tc5))...
        student_table(length(tc6)-1)*std(tc6)/sqrt(length(tc6))...
        student_table(12)*std(tc7)/sqrt(length(tc7))];

%partie graphique
figure;
set(gca,'yscale','log');% plot en échelle logarithnique
hold on;
plot( x, MoyTempQadr , '.-m', x , MoyTempGeom ,'.-r', x , MoyTempCour ,'.-b');
errorbar(x, MoyTempQadr, ErrTempQadr,'m');
errorbar(x, MoyTempGeom, ErrTempGeom,'r');
errorbar(x, MoyTempCour, ErrTempCour,'b');
hold off;
legend('err quad','err geom','err curb');
title('Temps d''exécution moyen vs nombre de sommets');
xlabel('Nombre de sommets');
ylabel('t(sec)');

figure;
hold on;
plot( x, MoyTempQadr , '.-m', x , MoyTempGeom ,'.-r', x , MoyTempCour ,'.-b');
errorbar(x, MoyTempQadr, ErrTempQadr,'.-m');
```

```
errorbar(x, MoyTempGeom, ErrTempGeom,'.-r');
errorbar(x, MoyTempCour, ErrTempCour,'.-b');
hold off;
title('Temps d''exécution moyen vs nombre de sommets');
legend('err quad','err geom','err curb');
xlabel('Nombre de sommets');
ylabel('t(sec)');
```

### B.2.1 Listing du programme Matlab, contenant les données des comparaisons numériques, qui affiche les graphiques des figures 6.4 à 6.9.

```
clear all;
format long e;

Stu_Const=1.644854;% constante de Student pour n=99 et alpha=0.05

%calcul de moyenne et d'intervalle de confiance des erreurs Emax, Emoy, Eqmoy de l'algorithme
%quadratique
devQ;
%calcul de moyenne et d'intervalle de confiance des erreurs Emax, Emoy, Eqmoy de l'algorithme
%géométrique
devG;
%calcul de moyenne et d'intervalle de confiance des erreurs Emax, Emoy, Eqmoy de l'algorithme
%de courbure
devC;

x=[0.05 , 0.1:0.1:0.9];% pourcentage de simplification

%partie graphique

figure;% erreur maximale

hold on;

plot(x, moy_Max_Q, ':.m',x,moy_Max_G,':.r',x,moy_Max_C,':.b');
errorbar(x, moy_Max_Q,zeros(size(x)),err_Max_Q,':.m');
errorbar(x, moy_Max_G,err_Max_G,':.r');
errorbar(x, moy_Max_C,err_Max_C,':.b');

hold off;

legend('err quad','err geom','err curb');
title('Erreur maximale absolue vs pourcentage de simplification');


figure;% erreur moyenne

hold on;

plot(x, moy_Moy_Q, ':.m',x,moy_Moy_G,':.r',x,moy_Moy_C,':.b');
errorbar(x, moy_Moy_Q,zeros(size(x)),err_Moy_Q,':.m');
errorbar(x, moy_Moy_G,err_Moy_G,':.r');
errorbar(x, moy_Moy_C,err_Moy_C,':.b');

hold off;
```

```
legend('err quad','err geom','err curb');
title('Erreur moyenne absolue vs pourcentage de simplification');


figure;% erreur quadratique moyenne

hold on;

plot(x, moy_QMoy_Q, ':.m',x,moy_QMoy_G,':.r',x,moy_QMoy_C,':.b');
errorbar(x, moy_QMoy_Q,zeros(size(x)),err_QMoy_Q,':.m');
errorbar(x, moy_QMoy_G,err_QMoy_G,':.r');
errorbar(x, moy_QMoy_C,err_QMoy_C,':.b');

hold off;

legend('err quad','err geom','err curb');
title('Erreur quadratique moyenne absolue vs pourcentage de simplification');
```

## B.2.2 Listing du programme « devQ.m » contenant les données des comparaisons numérique.

```
err_max95=[...
0.00106128 0.000723247 0.00138891 0.00181726 0.000601254 0.00253567 0.0015072 0.000496576 0.00364932 ...
0.00109681 0.0904205 0.00200043 0.00107186 0.000719126 0.00171741 0.000354035 0.00130746 0.00161455 ...
0.000980162 0.00162188 0.00105525 0.00125582 0.00184276 0.00382032 0.000805667 0.00902855 0.496037 ...
0.000411343 0.00263607 0.000969131 0.00285153 0.0109896 0.00182022 0.00253599 0.00126314 0.00296752 ...
0.00151982 0.000946306 0.00174415 0.000967978 0.00172919 0.00124384 0.00238686 0.00118505 0.000297466 ...
0.00219433 0.00108186 0.0008245 0.00129071 0.000835277 0.00129111 0.000889976 0.00262788 0.000476202 ...
0.00807122 0.00997129 0.000537991 0.00928878 0.000414847 0.000862107 0.000478143 0.000858741 0.000524719 ...
0.000840961 0.00344302 0.001002 0.00129579 0.00439881 0.0012234 0.00366216 0.00040034 0.000896793 ...
0.00221914 0.000780996 0.00132631 0.00273532 0.000496022 0.00173059 0.00354239 0.00109048 0.00247309 ...
0.00246009 7.06222e-006 0.00126122 0.00220742 0.00488384 0.00211789 0.00273532 0.000496022 0.00173059 ...
0.00354239 0.00109048 0.00247309 0.000900931 0.000588396 0.000898427 0.00106285 0.000522832 0.000643678];
err_max90=[...
0.000447666 0.00158134 0.000357295 0.000560039 0.000303921 0.000452225 0.000631641 0.00115336 0.000587842 ...
0.000872255 0.0700764 0.000537032 0.000455855 0.00235746 0.000364824 0.00033044 0.000514675 0.000528673 ...
0.000733362 0.000540806 0.000486469 0.000817616 0.000632298 0.00141037 0.00130165 0.0104877 0.466663 ...
0.00135231 0.00096927 0.000454372 0.00105759 0.00973511 0.00194282 0.000690433 0.000486975 0.000376027 ...
0.000436517 0.00113821 0.00041883 0.000353368 0.000660437 0.000767301 0.000578713 0.00059185 0.000309154 ...
0.000872334 0.000439169 0.00131823 0.000384706 0.00046569 0.0013365 0.000581784 0.000658474 0.000476202 ...
0.00765227 0.00878156 0.00110272 0.00928584 0.00031569 0.00145175 7.75567e-005 0.0001256 0.00094672 ...
0.000252811 0.000765369 0.001002 0.000541591 0.000803329 0.00177728 0.00118713 0.00167148 0.000896793 ...
0.0005505 0.00136227 0.000572859 0.000671373 0.000350135 0.000796636 0.000692229 0.000409093 0.00144117 ...
0.000999667 7.06222e-006 0.000753472 0.000717715 0.00488384 0.00128054 0.000671373 0.000350135 0.000796636 ...
0.000692229 0.000409093 0.00144117 0.000422592 0.000971867 0.000525011 0.000663638 0.0016207 0.000303404];
err_max80=[...
9.19148e-005 0.00145112 0.000103298 0.000560039 3.52634e-005 0.000860028 0.000250464 0.000426157 0.000632208 ...
0.000304353 0.0419491 0.000359749 0.000117011 0.00235746 0.000267676 0.000262357 0.00027873 0.000482021 ...
0.000931549 0.000679221 0.000109322 0.000817616 0.000524672 0.000887498 0.000775871 0.00900327 0.402656 ...
0.000767857 0.000199317 0.00076793 0.00054869 0.00814508 0.000999082 0.000441681 0.000848475 0.000332122 ...
0.000423361 0.000195165 0.000258821 0.00023941 0.000146318 0.000957011 0.000313362 0.000210705 0.000297466 ...
0.00032572 0.000428183 0.000281235 0.000136899 0.000200423 0.00146176 0.000154946 0.000894429 0.000546898 ...
0.00766423 0.00505357 0.000675985 0.00928584 6.93125e-005 0.00132063 0.000159645 2.41759e-006 0.000429898 ...
8.83579e-005 0.000491616 0.000668145 0.000389582 0.000359462 0.00164685 0.000414858 1.93856e-005 0.000897657 ...
0.000221957 0.000839455 0.000283519 0.000112877 0.000244422 0.000892993 0.000165611 0.000726979 0.000278415 ...
0.000304603 6.17944e-006 0.000350369 0.00067577 0.00276991 0.000207834 0.000112877 0.000244422 0.000892993 ...
0.000165611 0.000726979 0.000278415 0.000296085 0.00123939 0.000337319 0.000194754 0.00139908 0.000304811];
err_max70=[...
8.67491e-005 0.000683847 0.000404196 0.000179573 0.000101396 0.00040718 0.000245606 0.000634354 0.000277837 ...
0.000304353 0.0297837 0.000816378 0.000117011 0.00235746 0.000473077 0.000262357 0.000704408 0.000406461 ...
0.000728341 0.000252786 0.000108741 0.00115439 0.000315922 0.000950673 0.00077583 0.00454015 0.292725 ...
0.000791207 0.000334977 0.000346775 0.000922182 0.00618876 0.0012737 0.000581967 0.000756793 0.000193586 ...
0.000423361 0.00044882 0.000245943 0.000241263 0.000110167 0.000276621 0.000313362 0.000119429 4.65661e-010 ...
0.00080589 0.000156804 2.97034e-007 0.000113875 0.000246403 0.000707006 6.01902e-005 0.000512364 0.000476202 ...
0.00766878 0.00453268 0.000155471 0.00836218 7.56259e-005 0.000366892 3.10693e-006 1.99657e-006 0.000485085 ...
0.000101768 0.000492388 0.000844011 0.000176996 0.000131939 0.000470571 7.2106e-005 1.07947e-005 0.000889853 ...
0.000643701 0.00131812 0.000192968 0.000138802 0.000207961 0.00097636 0.000201737 0.000651611 0.0010162 ...
0.000567381 6.14106e-006 0.000350369 0.000479622 0.00435276 0.000277261 0.000138802 0.000207961 0.00097636 ...
0.000201737 0.000651611 0.0010162 0.000247013 0.00121524 0.000304522 9.34875e-005 0.00139908 0.000282093];
err_max60=[...
0.0001576 0.000623149 0.000404196 0.000179573 0.000333438 0.000750275 0.00020471 0.00035887 0.000175573 ...
0.000304353 0.0297665 2.87388e-005 0.000117011 0.00235746 0.000140404 0.000262357 0.000420658 0.000358682 ...
0.000712428 0.00030383 0.000108741 0.000464909 5.45e-005 7.31653e-005 0.00077583 0.00334584 0.139505 ...
0.000791207 0.000412245 0.000205555 0.000997732 0.000791422 0.00151386 0.000342697 0.0010228 0.000245106 ...
0.000281134 0.000195165 0.000201456 0.00013054 0.000113616 0.000259701 0.000383253 0.000117426 0.000297466 ...
0.000159914 0.000104434 0.00112152 0.000111165 0.000246403 0.00052155 6.71523e-005 0.00104735 0.00056934 ...
0.00766878 0.00409751 0.000392824 0.00836218 0.000176995 0.000814137 1.07735e-006 2.21307e-006 0.00037468 ...
1.55469e-006 0.000212847 9.31323e-010 8.40028e-005 1.87031e-005 0.00109957 0.000254176 0.000398794 0.000632946 ...
4.65502e-005 0.000540956 0.000557694 6.33874e-005 0.000355701 0.00097636 0.000339143 0.000993518 0.000101431 ...
0.000569948 6.09919e-006 0.000250806 0.000795768 0.00435276 0.000499391 6.33874e-005 0.000355701 0.00097636 ...
0.000339143 0.000993518 0.000101431 0.000261668 0.00100085 0.000304522 9.34875e-005 0.00100343 0.000255571];
```

```
err_max50=[...
5.92433e-005 0.000665006 0.000344092 0.000103052 0.000333438 0.000427101 0.00020471 0.000134289 0.000383295 ...
0.000304353 0.0296915 0.000312571 0.000109501 0.00140332 0.000120697 0.00040207 0.000456277 0.000196186 ...
0.000772223 0.000456463 0.000108741 9.15099e-005 0.000370591 0.000474006 0.000703477 0.00334584 0.0382297 ...
0.000960479 0.000192914 0.000504173 0.000777978 0.00035376 0.00143404 0.000372551 0.000483038 0.000245106 ...
0.000230195 0.000182215 0.000519521 3.46991e-005 4.6365e-005 0.00017836 0.000423644 0.000164465 4.65661e-010 ...
0.000220504 0.000140352 0.000436352 0.000192024 8.77313e-005 0.000527478 1.20493e-005 0.00104735 0.0003369 ...
0.00210702 0.00409751 0.00125867 0.00836218 2.38113e-005 0.00105678 3.98628e-005 1.96894e-006 0.000447137 ...
1.81131e-005 7.34126e-005 0.000526714 6.30021e-005 3.22054e-005 0.00105481 1.8593e-005 0.000213323 0.000632946 ...
9.80477e-005 0.000336439 0.000557694 0.000241248 4.65661e-010 0.00095815 0.000178756 0.000368906 0.000309939 ...
0.000553864 5.13616e-006 0.000346254 0.000414221 0.00501835 0.000609076 0.000241248 4.65661e-010 0.00095815 ...
0.000178756 0.000368906 0.000309939 0.000316717 0.000518799 0.000304522 0.000202275 0.00139908 0.000237429];
err_max40=[...
0.0001576 0.000113549 0.000346749 0.000560872 5.17183e-005 0.000427101 0.000121647 3.032e-007 0.000154256 ...
0.000304353 0.0298486 0.000108354 0.00010401 0.000377001 0.000335895 2.82866e-005 0.000456277 0.000196186 ...
0.000283598 0.00023823 9.07481e-005 0.000464909 0.000370591 0.000474006 0.000850289 0.00334584 0.00385541 ...
0.000540976 0.000225999 7.59959e-007 0.000777978 0.000332437 0.000499136 8.37127e-005 0.000727457 0.000245106 ...
0.000230195 0.000150065 0.000371316 3.41733e-005 4.41571e-005 0.000553616 0.000423644 8.00014e-005 0.000484205 ...
0.000548885 0.000148229 2.97859e-008 0.000193253 8.77313e-005 0.000271993 7.62743e-006 0.00018879 0.000287689 ...
0.00266264 0.00409751 0.000761337 0.00801436 2.28363e-005 0.00116169 4.52535e-007 2.21307e-006 0.000429898 ...
1.27636e-006 0.000111368 9.31323e-010 8.40028e-005 0.00025537 1.44549e-005 3.31364e-005 0.000855113 0.00060733 ...
0.000244573 0.00124532 5.55068e-005 0.00037297 0.00021022 0.000161218 0.000325671 0.000850188 0.000178833 ...
0.000432942 3.53111e-006 6.04119e-005 0.000136117 0.00501835 0.000550966 0.00037297 0.00021022 0.000161218 ...
0.000325671 0.000850188 0.000178833 0.000316717 0.000775489 0.000337319 9.34875e-005 8.06549e-010 0.000230526];
err_max30=[...
4.80004e-005 0.000418747 0.000404196 0.0001576 0.000102152 0.000384421 0.000421499 8.06549e-010 0.000154256 ...
0.000269056 0.0287827 1.03631e-005 0.000253645 0.000377001 0.000335895 3.55916e-005 0.000314467 0.000397332 ...
0.000391136 0.00023823 3.44962e-005 0.000464909 7.80558e-006 4.16736e-005 7.36275e-010 0.00292721 0.00217302 ...
0.000540976 1.40136e-005 7.59959e-007 0.000777978 0.000332437 0.000426098 9.43259e-005 0.000632111 7.43334e-005 ...
0.000230195 0.000149625 0.000152118 0.000407206 0.000485781 2.3899e-007 0.000423644 8.00014e-005 4.65661e-010 ...
0.000548885 0.000138119 1.98573e-008 0.000185804 8.77313e-005 4.25917e-005 6.75253e-005 3.94684e-005 5.83672e-005 ...
0.00264565 0.00409751 8.06549e-010 0.00805659 4.67127e-006 0.000636592 4.08604e-007 2.21307e-006 0.000429898 ...
9.15883e-007 0.000938717 0.000304908 0.000131714 1.48585e-005 5.06639e-007 1.44826e-005 0.000684779 1.75311e-005 ...
0.000244573 6.65585e-007 0.000463752 6.17257e-006 3.29272e-010 5.76925e-005 5.02455e-005 6.41152e-005 7.46541e-005 ...
0.000167107 3.52607e-006 6.04406e-005 0.00065163 0.00466699 0.000290978 6.17257e-006 3.29272e-010 5.76925e-005 ...
5.02455e-005 6.41152e-005 7.46541e-005 0.000316717 0.000557596 0.000304522 9.34875e-005 0.000129266 0.000230526];
err_max20=[...
0.000269617 0.000123394 0.000346749 2.95896e-006 0.000102152 0.000291396 4.53039e-005 3.032e-007 0.000154256 ...
0.000270709 0.00400721 0.000598913 1.29331e-005 9.31323e-010 3.40964e-006 0.000131255 8.80298e-005 0.000397332 ...
1.45746e-005 2.75861e-006 3.6483e-006 1.14063e-009 0.000370591 0.000474006 2.07503e-005 0.00307434 0.00122443 ...
0.000540976 9.49671e-006 7.59959e-007 0.000777978 0.000332437 0.000255264 3.43873e-005 0.000829193 0.00010574 ...
3.20077e-005 0.00010843 0.000779874 3.50305e-005 1.16415e-010 8.06549e-010 3.84594e-005 1.83092e-005 0.000144933 ...
0.000220504 6.9182e-005 2.97859e-008 3.89765e-005 8.77313e-005 3.39263e-005 0.000104342 1.15848e-008 1.6861e-005 ...
0.000945094 0.00409751 8.06549e-010 0.00801068 4.5521e-006 0.000175075 1.7708e-005 1.09473e-006 8.06549e-010 ...
1.7708e-005 0.000521265 1.31709e-009 3.28465e-006 5.443e-006 0.00023447 1.8593e-005 9.66883e-006 1.33153e-005 ...
7.49754e-006 0.000107325 0.000169975 6.17231e-006 2.32831e-010 4.86211e-005 3.9746e-005 8.06549e-010 7.4701e-005 ...
0.000203079 5.26836e-009 3.5855e-005 0.000365159 0.00458998 0.000290978 6.17231e-006 2.32831e-010 4.86211e-005 ...
3.9746e-005 8.06549e-010 7.4701e-005 0.000316717 6.58545e-010 0.000304522 0.000154891 9.31323e-010 0.000242107];
err_max10=[...
2.57797e-006 9.31323e-010 0.00033122 8.4512e-005 3.29272e-010 0.000274812 1.16415e-010 9.31323e-010 0.000154256 ...
0.00011077 6.44563e-007 7.96504e-006 0.000352347 9.31323e-010 0.000244493 6.29027e-005 7.8608e-005 0.000196186 ...
0.000526184 0.000533801 2.70375e-006 1.14063e-009 2.83753e-006 4.28643e-005 1.89153e-009 0.00172697 4.18627e-009 ...
0.00045347 4.11147e-006 8.06549e-010 0.000777978 8.43308e-005 1.31709e-009 3.43873e-005 8.06549e-010 4.97182e-006 ...
3.34034e-006 4.65661e-010 7.72343e-005 1.16415e-010 1.16415e-010 8.06549e-010 7.77255e-006 0.000116796 2.14887e-005 ...
0.00125124 4.08177e-007 6.58545e-010 1.20269e-007 2.0952e-006 8.06549e-010 1.20269e-007 0 3.29272e-010 ...
0.00094161 0.00443093 8.06549e-010 1.16415e-010 4.41544e-006 8.06549e-010 1.31779e-007 0 0.000429898 ...
3.95255e-007 0.000159885 9.31323e-010 1.0922e-005 5.443e-006 9.31323e-010 1.44826e-005 1.41222e-006 8.13792e-005 ...
3.78494e-006 8.06549e-010 1.64636e-010 6.17231e-006 2.32831e-010 5.17321e-005 4.09782e-005 0.000565923 7.46541e-005 ...
2.4724e-005 6.45239e-009 0.000162716 0.000567926 0.00533455 7.20024e-006 6.17231e-006 2.32831e-010 5.17321e-005 ...
4.09782e-005 0.000565923 7.46541e-005 0.000279336 0.000184897 0.000304522 9.68075e-005 9.31323e-010 0.000270317];
```

```
% calcul de l'intervalle de confiance selon la distribution de Student
err_Max_Q=[Stu_Const*std(err_max95)/sqrt(length(err_max95)) ...
          Stu_Const*std(err_max90)/sqrt(length(err_max90)) ...
          Stu_Const*std(err_max80)/sqrt(length(err_max80)) ...
          Stu_Const*std(err_max70)/sqrt(length(err_max70)) ...
          Stu_Const*std(err_max60)/sqrt(length(err_max60)) ...
          Stu_Const*std(err_max50)/sqrt(length(err_max50)) ...
          Stu_Const*std(err_max40)/sqrt(length(err_max50)) ...
          Stu_Const*std(err_max30)/sqrt(length(err_max40)) ...
          Stu_Const*std(err_max20)/sqrt(length(err_max20)) ...
          Stu_Const*std(err_max10)/sqrt(length(err_max10)) ];

moy_Max_Q=[mean(err_max95) mean(err_max90) mean(err_max80) mean(err_max70)...
mean(err_max60) mean(err_max50) mean(err_max40) mean(err_max30) mean(err_max20)...
mean(err_max10) ];%calcul des moyennes

err_moy95=[...
5.05869e-005 3.09171e-007 0.000131743 5.12335e-005 1.4045e-007 0.00012622 9.37853e-005 1.62704e-007 0.000143726 ...
7.45793e-005 0.00314749 0.000135202 4.97772e-005 2.01785e-007 5.19059e-005 6.87809e-006 3.23624e-007 3.4612e-005 ...
3.06515e-007 6.67947e-005 5.10437e-005 3.25179e-007 4.47453e-005 0.000121367 3.23911e-007 0.000160015 0.096161 ...
6.38698e-008 0.000364753 1.58669e-007 0.000222359 0.000144967 3.28614e-007 0.00021327 2.57964e-007 0.000119088 ...
3.31814e-005 2.42905e-007 7.49903e-005 4.78229e-005 9.91139e-005 1.73164e-007 0.000179845 0.000103216 5.07038e-008 ...
0.000149279 5.56483e-005 1.65617e-007 4.25688e-005 3.08119e-005 4.71409e-007 3.69466e-005 3.97697e-005 1.0994e-007 ...
7.09867e-005 8.03798e-005 1.13955e-007 0.000103784 2.95828e-005 3.481e-007 2.1179e-005 5.70852e-005 1.65877e-007 ...
2.56513e-005 8.23167e-005 1.47412e-007 9.20047e-005 8.9222e-005 1.90972e-007 6.9598e-005 5.28349e-008 7.34225e-006 ...
0.000120421 3.02895e-007 6.2965e-005 0.000118579 6.93238e-008 0.000156148 0.000359387 3.71554e-007 0.000204846 ...
4.307e-005 6.46828e-007 6.45871e-005 0.000141844 3.28552e-006 5.89147e-005 0.000118579 6.93238e-008 0.000156148 ...
0.000359387 3.71554e-007 0.000204846 3.12775e-005 1.20553e-007 2.67329e-005 4.00829e-005 9.03661e-008 2.56464e-005 ];
err_moy90=[...
1.08241e-005 3.55459e-007 1.63833e-005 1.76076e-005 6.556e-008 2.65e-005 2.78889e-005 2.99385e-007 1.48937e-005 ...
1.85983e-005 0.00155971 6.02978e-005 1.2463e-005 2.67875e-007 5.56935e-006 4.71845e-006 1.17592e-007 4.74432e-006 ...
1.81957e-007 4.70051e-006 1.09651e-005 2.28866e-007 3.17003e-006 3.745e-005 3.56227e-007 5.77462e-005 0.0854999 ...
2.00245e-007 4.8122e-005 1.93983e-007 6.54554e-005 5.57405e-005 3.38538e-007 1.8227e-005 9.38909e-008 1.61405e-005 ...
1.09402e-005 1.92906e-007 4.20426e-005 1.27151e-005 1.74412e-005 2.46596e-007 3.80055e-005 1.95948e-005 8.337e-008 ...
3.58609e-005 1.65918e-005 2.92706e-007 9.70195e-006 8.70454e-006 3.54434e-007 9.95202e-006 3.02034e-006 6.29282e-008 ...
3.84232e-005 2.65072e-005 3.27748e-007 5.66068e-005 8.82606e-006 5.665e-007 4.34807e-006 4.77546e-006 3.55032e-007 ...
7.60422e-006 1.27557e-005 2.9287e-007 1.39932e-005 1.21395e-005 2.7503e-007 1.50493e-005 3.85621e-007 8.50113e-006 ...
1.32326e-005 4.84961e-007 1.29618e-005 5.68301e-005 8.54074e-008 6.54467e-005 4.96827e-005 8.33538e-008 7.71374e-005 ...
1.09723e-005 6.41885e-007 1.33266e-005 2.05301e-005 3.30867e-006 1.32135e-005 5.68301e-005 8.54074e-008 6.54467e-005 ...
4.96827e-005 8.33538e-008 7.71374e-005 1.11798e-005 1.11035e-007 4.83073e-006 1.11867e-005 5.64691e-007 5.65285e-006 ];
err_moy80=[...
4.9885e-006 3.6007e-007 4.67779e-006 1.20026e-005 2.30678e-008 5.62437e-006 1.5284e-006 8.59648e-008 3.53016e-006 ...
1.82929e-006 0.000343506 1.45783e-005 2.34955e-006 3.42121e-007 9.9238e-007 1.55532e-006 7.78383e-008 1.0383e-006 ...
1.47287e-007 1.23088e-006 1.97778e-006 1.98094e-007 1.17915e-006 4.41488e-006 1.54733e-007 2.0628e-005 0.0539968 ...
2.90299e-007 2.355e-006 1.6837e-007 3.27643e-006 2.50719e-005 1.65827e-007 2.74642e-006 1.35078e-007 4.63456e-006 ...
3.50667e-006 1.01204e-007 1.19815e-005 7.44694e-006 2.61859e-006 1.89445e-007 4.89792e-006 8.34014e-006 7.27527e-008 ...
5.38131e-006 2.00518e-006 4.62924e-008 1.29226e-006 1.5914e-006 2.7769e-007 9.8796e-007 1.18953e-006 1.49105e-007 ...
3.42324e-005 6.581e-006 1.60587e-007 4.94461e-005 1.25057e-006 2.35101e-007 3.65084e-007 1.4996e-007 1.21556e-007 ...
3.59967e-007 1.45842e-006 2.26156e-007 3.09416e-006 2.47525e-006 1.94351e-007 1.54473e-006 2.70743e-009 5.87571e-006 ...
1.23373e-006 1.8824e-007 2.88705e-006 6.70481e-006 3.27013e-008 1.16816e-005 6.99649e-006 9.58704e-008 1.03331e-005 ...
3.90193e-006 2.52041e-007 1.09104e-005 3.69185e-006 2.00296e-006 4.02874e-006 6.70481e-006 3.27013e-008 1.16816e-005 ...
6.99649e-006 9.58704e-008 1.03331e-005 1.28766e-006 1.78062e-007 1.47888e-006 9.60682e-007 3.67626e-007 2.3814e-006 ];
err_moy70=[...
7.29474e-007 3.67461e-008 1.98428e-006 2.20136e-006 4.54355e-008 8.47718e-007 1.17959e-006 1.15523e-007 1.09572e-006 ...
1.41325e-006 0.000146626 5.13384e-006 1.71221e-006 4.65209e-007 5.90521e-007 1.13559e-006 1.15121e-007 6.97557e-007 ...
5.02231e-008 5.40149e-007 1.07002e-006 3.27771e-007 5.88193e-007 2.16703e-006 6.40767e-008 6.60449e-006 0.032132 ...
3.84465e-007 8.35839e-007 5.56232e-008 1.3382e-006 1.36449e-005 2.14728e-007 7.48647e-007 2.36184e-007 2.16708e-006 ...
2.91856e-006 1.8735e-007 6.424e-006 5.58488e-006 1.82162e-006 5.87715e-008 2.59612e-006 5.15358e-006 9.0597e-013 ...
2.54007e-006 3.8231e-007 4.1331e-011 4.84945e-007 4.79992e-007 2.0814e-007 5.09301e-007 3.56261e-007 1.14108e-007 ...
3.0052e-005 3.13344e-006 1.41297e-008 3.83718e-005 3.96046e-007 5.31042e-008 1.02872e-007 4.31557e-008 1.07107e-007 ...
```

1.41042e-007 1.09853e-006 1.59307e-007 1.34195e-006 8.9295e-007 1.08786e-007 9.44763e-007 1.46022e-009 2.50293e-006 ...
1.08542e-006 1.46238e-007 1.60354e-006 6.60206e-007 3.42878e-008 6.42664e-006 1.91052e-006 6.95472e-008 6.28172e-006 ...
1.09143e-006 1.36739e-007 5.00888e-006 1.74376e-006 2.93672e-006 2.46156e-006 6.60206e-007 3.42878e-008 6.42664e-006 ...
1.91052e-006 6.95472e-008 6.28172e-006 7.24052e-007 2.77621e-007 1.3676e-006 5.90199e-007 4.07137e-007 2.02649e-006 ];
err_moy60=[...
4.85468e-007 1.44552e-007 1.3979e-006 2.05321e-006 6.84131e-008 5.9808e-007 7.35606e-007 3.2719e-008 5.76279e-007 ...
9.9676e-007 0.000134616 1.90891e-006 1.50896e-006 3.0636e-007 4.21185e-007 6.85143e-007 1.0966e-007 3.41416e-007 ...
1.43049e-007 3.91699e-007 1.27795e-006 1.22082e-007 3.88268e-007 1.28082e-006 1.24858e-007 2.84962e-006 0.00589188 ...
2.24357e-007 5.82501e-007 1.98838e-008 1.10899e-006 1.99829e-006 4.52403e-007 5.88717e-007 1.58057e-007 1.54975e-006 ...
1.02942e-006 1.17005e-007 3.10543e-006 2.07665e-006 1.45575e-006 3.20399e-008 1.65503e-006 4.66144e-006 3.06325e-008 ...
1.14724e-006 1.84493e-007 3.38112e-008 2.92075e-007 2.95419e-007 7.91632e-008 2.60374e-007 2.82913e-007 9.51633e-008 ...
2.95419e-005 2.36622e-006 5.66095e-008 3.74497e-005 2.44843e-007 1.53293e-007 4.44407e-008 2.83456e-008 4.67495e-008 ...
6.04785e-008 7.04154e-007 5.1572e-012 7.86981e-007 5.32263e-007 1.25635e-007 7.10205e-007 2.49754e-008 1.92079e-006 ...
6.08815e-007 6.13588e-008 1.04803e-006 2.32826e-007 1.20161e-007 3.23058e-006 7.08695e-007 1.94772e-007 4.62399e-006 ...
9.65952e-007 1.24651e-007 3.45778e-006 1.57656e-006 3.08205e-006 1.47326e-006 2.32826e-007 1.20161e-007 3.23058e-006 ...
7.08695e-007 1.94772e-007 4.62399e-006 6.50727e-007 2.53219e-007 1.12825e-006 3.88849e-007 1.37744e-007 1.54567e-006 ];
err_moy50=[...
1.12398e-006 4.45151e-008 1.22946e-006 1.01076e-006 4.99524e-008 5.13428e-007 6.27002e-007 1.72267e-008 4.61624e-007 ...
8.07959e-007 0.000139924 8.06052e-007 8.85106e-007 3.86716e-007 3.98784e-007 3.52893e-007 1.33057e-007 2.27739e-007 ...
1.4788e-007 3.16189e-007 1.63171e-006 1.08345e-008 3.76636e-007 9.81823e-007 1.04699e-007 2.82708e-006 0.00057525 ...
3.57647e-007 3.09727e-007 2.42991e-008 6.53412e-007 1.33195e-006 3.9312e-007 4.43477e-007 1.19025e-007 1.57394e-006 ...
3.94225e-007 5.85602e-008 2.57243e-006 1.58358e-006 4.03067e-007 1.87226e-008 1.80709e-006 2.8682e-006 2.23111e-012 ...
1.00064e-006 1.49235e-007 4.56377e-008 1.75518e-007 4.06251e-007 4.79434e-008 5.46891e-008 1.07025e-007 5.14986e-008 ...
3.69206e-006 2.40497e-006 1.80156e-007 3.71844e-005 1.37957e-007 8.16814e-008 3.03298e-008 1.99227e-008 9.50617e-008 ...
4.07246e-008 4.40042e-007 1.02881e-007 3.14909e-007 4.10306e-007 2.02743e-007 3.79942e-007 3.5184e-008 9.48376e-007 ...
3.98294e-007 3.23585e-008 8.42373e-007 1.15876e-007 2.45405e-012 1.41785e-006 5.0093e-007 1.03439e-007 2.82361e-006 ...
5.31886e-007 4.41763e-008 3.64635e-006 8.76979e-007 3.96928e-006 9.76031e-007 1.15876e-007 2.45405e-012 1.41785e-006 ...
5.0093e-007 1.03439e-007 2.82361e-006 5.47686e-007 1.22456e-007 7.53948e-007 3.55213e-007 1.63509e-007 1.27925e-006];
err_moy40=[...
7.24506e-007 1.81805e-008 1.18501e-006 8.48598e-007 1.81538e-008 4.77888e-007 5.12343e-007 4.51848e-011 3.113e-007 ...
7.74318e-007 0.000149667 3.62335e-007 4.72831e-007 2.99382e-008 4.0107e-007 2.73514e-007 9.19606e-008 1.97782e-007 ...
2.71801e-008 2.68637e-007 1.0987e-006 4.58557e-008 3.68855e-007 7.19784e-007 8.35378e-008 2.7649e-006 3.18214e-006 ...
1.40848e-007 2.00399e-007 1.00599e-010 4.87823e-007 9.33643e-007 5.51151e-008 1.89269e-007 1.30342e-007 1.55912e-006 ...
3.0682e-007 2.97735e-008 2.25411e-006 1.39281e-006 2.95136e-007 1.25384e-007 6.05507e-007 1.15655e-006 7.55276e-008 ...
8.87341e-007 9.62162e-008 1.71915e-011 6.12606e-008 2.84147e-007 3.1332e-008 2.38183e-008 4.26505e-008 2.76416e-008 ...
4.48463e-006 2.35496e-006 1.30316e-007 3.29826e-005 1.11751e-007 6.03596e-008 8.68387e-009 1.27037e-008 1.24768e-007 ...
2.46081e-008 3.09779e-007 1.30891e-011 2.01172e-007 3.44124e-007 1.06289e-009 2.4129e-007 1.11368e-007 8.62492e-007 ...
3.85421e-007 9.38377e-008 9.84842e-008 8.41769e-008 3.41728e-008 9.53457e-007 4.79312e-007 5.53134e-008 2.23638e-006 ...
4.59269e-007 9.73802e-009 2.37751e-006 8.18882e-007 3.40956e-006 5.00174e-007 8.41769e-008 3.41728e-008 9.53457e-007 ...
4.79312e-007 5.53134e-008 2.23638e-006 4.94598e-007 5.3678e-008 6.93512e-007 3.02024e-007 8.42262e-012 9.5798e-007 ];
err_moy30=[...
7.20394e-008 1.20086e-007 1.17093e-006 1.58274e-008 2.36879e-008 4.45698e-007 5.6477e-007 1.099e-011 3.05955e-007 ...
6.47553e-007 0.000159452 2.16138e-007 2.80098e-007 5.78096e-008 3.99465e-007 2.32892e-007 4.13165e-008 2.43037e-007 ...
7.00866e-008 2.6e-007 1.33052e-008 4.29841e-008 3.00779e-007 4.2789e-007 1.23712e-011 1.8807e-006 8.71155e-007 ...
2.20847e-007 1.14318e-007 1.00864e-010 4.60527e-007 5.69026e-007 2.89629e-008 1.52889e-007 6.54274e-008 1.53483e-006 ...
2.76373e-007 3.80364e-008 2.19621e-006 1.35981e-006 3.65348e-007 2.30251e-011 5.56921e-007 1.15446e-006 4.0165e-012 ...
9.12546e-007 3.19299e-008 1.62619e-011 1.6597e-007 2.76285e-007 2.40465e-009 1.75879e-008 5.49088e-009 4.47905e-009 ...
8.37739e-006 2.34082e-006 8.63355e-012 2.41935e-005 8.525e-008 1.19363e-007 2.97279e-009 1.12407e-008 2.49969e-008 ...
9.91139e-009 3.86996e-007 2.18107e-008 2.49754e-008 2.36105e-007 1.58868e-010 1.54469e-007 7.08222e-008 3.88115e-007 ...
3.51545e-007 4.82515e-010 8.56447e-008 2.39891e-008 2.00082e-012 5.48711e-007 4.04635e-007 4.0147e-009 1.86755e-006 ...
2.90726e-007 2.35003e-009 1.58307e-006 8.81283e-007 3.45413e-006 3.01486e-007 2.39891e-008 2.00082e-012 5.48711e-007 ...
4.04635e-007 4.0147e-009 1.86755e-006 4.68992e-007 7.38381e-008 5.60609e-007 2.81953e-007 1.16888e-008 6.54131e-007 ];
err_moy20=[...
4.44568e-008 1.08882e-008 1.07486e-006 3.71861e-009 1.9988e-008 3.72201e-007 3.51384e-007 4.42079e-011 2.83861e-007 ...
9.26427e-008 7.80621e-007 2.29988e-007 1.82057e-008 1.12148e-011 2.42885e-007 2.62091e-008 9.22703e-009 2.36441e-007 ...
1.44786e-009 1.56764e-007 3.73655e-009 2.88519e-011 2.96217e-007 4.64666e-007 1.73441e-009 3.87117e-006 5.8227e-007 ...
1.01872e-007 9.93545e-008 1.00614e-010 4.49182e-007 5.22598e-007 1.60997e-008 1.28744e-007 5.86794e-008 1.16685e-006 ...
2.07825e-007 8.33032e-009 1.2756e-006 1.02303e-006 1.99549e-013 7.88336e-012 5.06444e-007 2.36815e-009 1.4051e-008 ...
8.59801e-007 1.24011e-008 1.41327e-011 9.50562e-009 2.71971e-007 3.79507e-009 2.08907e-008 2.79084e-012 2.34556e-009 ...
1.4602e-006 2.36336e-006 8.33676e-012 2.24908e-005 6.07007e-008 1.72661e-008 6.87892e-009 3.8484e-009 7.29341e-012 ...
4.41619e-009 2.59684e-007 2.21588e-011 7.21738e-010 1.26364e-007 3.63926e-008 5.66819e-008 9.21774e-010 3.74095e-008 ...
1.97645e-007 9.99754e-009 3.87758e-008 4.03057e-009 1.49325e-012 4.58066e-007 3.97576e-007 8.97401e-012 1.79302e-006 ...
1.69561e-007 4.11124e-011 1.50114e-006 8.36698e-007 2.84289e-006 3.13469e-007 4.03057e-009 1.49325e-012 4.58066e-007 ...
3.97576e-007 8.97401e-012 1.79302e-006 5.24864e-007 8.04555e-012 5.59475e-007 2.9988e-007 8.53866e-012 6.82878e-007 ];

```
err_moy10=[...
1.22446e-009 8.338e-012 1.62033e-007 1.31911e-008 6.06926e-012 1.31453e-007 6.91454e-013 8.71617e-012 2.69246e-007 ...
1.2925e-008 2.82648e-010 1.25851e-007 3.45271e-008 1.18988e-011 3.11735e-007 8.91758e-009 2.67894e-009 8.66401e-008 ...
3.28947e-008 2.67931e-007 2.97784e-009 3.18198e-011 9.89939e-008 3.76907e-007 1.31345e-011 8.89329e-007 1.84438e-011 ...
3.51996e-008 6.85157e-008 1.12223e-011 3.18129e-007 2.58086e-007 3.05924e-011 8.76152e-008 8.4458e-012 2.23103e-007 ...
1.26032e-008 7.1013e-012 1.90447e-007 3.79873e-013 2.82825e-013 7.91492e-012 2.24569e-007 7.46514e-009 1.47913e-009 ...
4.92932e-007 4.46937e-010 1.2215e-011 1.21268e-010 3.63626e-008 1.15006e-011 2.11846e-010 0 5.76681e-012 ...
7.31579e-007 2.22223e-006 7.41957e-012 7.8431e-014 5.11818e-008 1.22782e-011 8.28686e-011 0 3.97541e-008 ...
8.50099e-010 1.07798e-007 2.63074e-011 1.44429e-009 7.23086e-008 1.15541e-011 5.57672e-008 1.33302e-010 5.35142e-008 ...
6.30173e-008 1.18621e-011 2.97597e-013 1.24386e-009 1.33753e-012 4.46281e-007 3.54684e-007 3.7192e-008 1.82045e-006 ...
1.54812e-007 4.3906e-011 4.78023e-007 6.09105e-007 2.3002e-006 1.59458e-007 1.24386e-009 1.33753e-012 4.46281e-007 ...
3.54684e-007 3.7192e-008 1.82045e-006 5.1772e-007 6.02061e-009 6.36277e-007 1.74622e-007 8.2035e-012 3.51925e-007];
```

```
% calcul de l'intervalle de confiance selon la distribution de Student
err_Moy_Q=[Stu_Const*std(err_moy95)/sqrt(length(err_moy95)) ...
           Stu_Const*std(err_moy90)/sqrt(length(err_moy90)) ...
           Stu_Const*std(err_moy80)/sqrt(length(err_moy80)) ...
           Stu_Const*std(err_moy70)/sqrt(length(err_moy70)) ...
           Stu_Const*std(err_moy60)/sqrt(length(err_moy60)) ...
           Stu_Const*std(err_moy50)/sqrt(length(err_moy50)) ...
           Stu_Const*std(err_moy40)/sqrt(length(err_moy50)) ...
           Stu_Const*std(err_moy30)/sqrt(length(err_moy40)) ...
           Stu_Const*std(err_moy20)/sqrt(length(err_moy20)) ...
           Stu_Const*std(err_moy10)/sqrt(length(err_moy10)) ];
moy_Moy_Q=[mean(err_moy95) mean(err_moy90) mean(err_moy80)...
 mean(err_moy70) mean(err_moy60) mean(err_moy50) mean(err_moy40)...
 mean(err_moy30) mean(err_moy20) mean(err_moy10) ];%calcul des moyennes
```

```
err_qmoy95=[...
0.00010891 8.53782e-006 0.000179265 0.000101548 4.92331e-006 0.000239096 0.000160261 5.27849e-006 0.00029195 ...
0.000145299 0.0119196 0.000219108 0.000106197 6.62719e-006 0.000148788 1.27727e-005 1.16488e-005 0.000106241 ...
9.44572e-006 0.000153463 0.000119928 1.11321e-005 0.000125457 0.000223108 9.79704e-006 0.000361593 0.159201 ...
3.48096e-006 0.000496823 7.0803e-006 0.000381361 0.000468686 1.30216e-005 0.00033116 9.72577e-006 0.000204975 ...
8.13034e-005 7.96692e-006 0.000131492 0.000110517 0.000187157 8.38485e-006 0.000270011 0.000177154 2.57126e-006 ...
0.000233012 0.000108358 6.61143e-006 9.57324e-005 7.12109e-005 1.45378e-005 7.77564e-005 0.000124019 4.70023e-006 ...
0.000340762 0.000346784 5.17012e-006 0.000459443 6.10184e-005 9.97189e-006 4.09955e-005 0.000105304 5.58797e-006 ...
5.91391e-005 0.00022473 7.95692e-006 0.000162933 0.000231941 9.12637e-006 0.000208686 2.66101e-006 2.50057e-005 ...
0.000286611 8.13292e-006 0.000135337 0.000175043 3.84521e-006 0.000236336 0.00057069 1.1377e-005 0.000297346 ...
0.000105066 1.51052e-006 0.000127955 0.000244639 6.88974e-005 0.00013189 0.000175043 3.84521e-006 0.000236336 ...
0.00057069 1.1377e-005 0.000297346 6.96538e-005 4.90393e-006 6.93546e-005 9.42671e-005 4.00561e-006 5.31412e-005 ];
err_qmoy90=[...
2.90958e-005 1.39809e-005 3.21725e-005 3.57314e-005 2.89504e-006 5.44043e-005 5.44992e-005 1.02867e-005 3.40726e-005 ...
4.71835e-005 0.00742372 8.99649e-005 4.08527e-005 1.50131e-005 1.85858e-005 9.48203e-006 4.59537e-006 2.03222e-005 ...
6.58376e-006 1.77058e-005 3.34279e-005 8.07665e-006 1.4837e-005 9.72531e-005 1.11279e-005 0.00031444 0.14455 ...
8.87545e-006 8.89099e-005 5.53916e-006 0.000153797 0.000364954 1.30088e-005 4.94191e-005 3.35651e-006 3.28312e-005 ...
3.01474e-005 7.7884e-006 6.66929e-005 3.11719e-005 5.20042e-005 7.21915e-006 6.76712e-005 5.06531e-005 3.22363e-006 ...
6.71393e-005 3.66983e-005 1.09909e-005 3.0523e-005 2.18853e-005 1.2191e-005 3.70705e-005 1.435e-005 3.39708e-006 ...
0.000311924 0.000282903 1.0433e-005 0.00044192 1.74344e-005 1.38367e-005 9.7051e-006 1.22274e-005 1.16006e-005 ...
1.65227e-005 3.77511e-005 9.72261e-006 4.27037e-005 4.17251e-005 1.11917e-005 4.74898e-005 1.5173e-005 2.49283e-005 ...
4.09686e-005 1.50042e-005 3.9665e-005 8.86748e-005 3.09597e-006 0.000125696 7.80735e-005 3.70968e-006 0.000146331 ...
3.26963e-005 1.50996e-006 3.52922e-005 4.0293e-005 7.3252e-005 4.27214e-005 8.86748e-005 3.09597e-006 0.000125696 ...
7.80735e-005 3.70968e-006 0.000146331 2.52571e-005 5.82965e-006 1.93899e-005 3.74875e-005 1.57939e-005 2.07111e-005 ];
```

```
err_qmoy80=[...
1.41258e-005 1.40917e-005 1.05944e-005 2.2424e-005 5.15524e-007 1.53261e-005 7.13179e-006 3.33554e-006 1.21723e-005 ...
```

9.43366e-006 0.00268958 2.28768e-005 9.36332e-006 1.58835e-005 4.57008e-006 4.32379e-006 2.82808e-006 7.09924e-006 ...
6.76893e-006 8.69351e-006 7.41209e-006 7.11376e-006 5.76907e-006 1.77806e-005 6.34743e-006 0.000255203 0.104976 ...
8.86812e-006 7.16627e-006 6.5957e-006 1.17427e-005 0.000274513 7.08773e-006 8.76202e-006 5.62526e-006 8.20586e-006 ...
1.23964e-005 2.62238e-006 2.64312e-005 1.75749e-005 1.06961e-005 6.81299e-006 1.38021e-005 2.35496e-005 3.07583e-006 ...
9.79917e-006 7.97328e-006 2.19396e-006 6.96682e-006 6.90055e-006 1.20153e-005 5.11325e-006 7.60197e-006 5.74348e-006 ...
0.000308355 0.000104769 6.18293e-006 0.000438845 2.40564e-006 1.08758e-005 1.91507e-006 3.409e-007 4.55028e-006 ...
1.28231e-006 5.30544e-006 8.08906e-006 1.16645e-005 1.00123e-005 1.12047e-005 5.06263e-006 1.38131e-007 2.13465e-005 ...
4.08216e-006 7.00893e-006 1.11325e-005 1.07471e-005 1.80064e-006 2.85555e-005 1.25564e-005 4.49427e-006 2.29267e-005 ...
8.4789e-006 8.19626e-007 2.15771e-005 9.90488e-006 4.17928e-005 1.04171e-005 1.07471e-005 1.80064e-006 2.85555e-005 ...
1.25564e-005 4.49427e-006 2.29267e-005 5.54154e-006 8.98426e-006 1.07685e-005 3.13865e-006 1.28319e-005 1.32708e-005];
err_qmoy70=[...
3.48366e-006 2.96509e-006 6.82452e-006 7.45523e-006 1.15387e-006 4.5352e-006 6.12695e-006 5.5202e-006 4.27968e-006 ...
8.68864e-006 0.00147791 9.98975e-006 8.0135e-006 1.77199e-005 5.99127e-006 3.28612e-006 5.18817e-006 5.07507e-006 ...
3.62791e-006 2.80852e-006 4.95394e-006 1.18198e-005 1.99223e-006 1.43351e-005 3.88328e-006 0.000106297 0.069256 ...
1.05836e-005 4.68898e-006 2.90179e-006 9.84508e-006 0.000177683 9.34735e-006 4.31878e-006 8.21513e-006 4.88199e-006 ...
1.18515e-005 4.85392e-006 1.26044e-005 1.40415e-005 8.53011e-006 2.53941e-006 6.09352e-006 1.47628e-005 8.96552e-012 ...
7.90472e-006 2.45616e-006 2.14612e-009 3.64256e-006 3.23457e-006 7.77986e-006 2.85359e-006 3.99176e-006 4.70131e-006 ...
0.000307583 7.19134e-005 1.01188e-006 0.000368717 1.31739e-006 2.66114e-006 1.84697e-007 1.69358e-007 4.68204e-006 ...
9.24349e-007 3.87445e-006 6.63092e-006 4.59832e-006 1.89897e-006 4.31158e-006 1.85441e-006 8.30297e-008 1.16895e-005 ...
4.47625e-006 8.861e-006 5.91576e-006 1.67028e-006 1.7356e-006 1.90133e-005 5.2951e-006 4.10988e-006 1.72091e-005 ...
6.06437e-006 6.22091e-007 1.22503e-005 7.03256e-006 6.03127e-005 7.6437e-006 1.67028e-006 1.7356e-006 1.90133e-005 ...
5.2951e-006 4.10988e-006 1.72091e-005 3.96989e-006 1.0641e-005 1.00013e-005 2.41003e-006 1.27591e-005 1.13622e-005];
err_qmoy60=[...
2.98563e-006 5.35204e-006 5.98852e-006 7.03738e-006 2.42369e-006 6.07592e-006 3.84366e-006 2.35666e-006 3.41761e-006 ...
8.17212e-006 0.00141766 3.60194e-006 7.5714e-006 1.644e-005 1.7416e-006 2.79842e-006 4.1733e-006 2.97012e-006 ...
6.23974e-006 3.01939e-006 5.05515e-006 4.34414e-006 9.02404e-007 4.79666e-006 5.28041e-006 5.67392e-005 0.0200565 ...
8.10297e-006 3.78516e-006 1.42597e-006 1.4656e-005 1.38048e-005 1.50826e-005 3.26534e-006 7.50735e-006 3.98253e-006 ...
5.12608e-006 3.15463e-006 6.86262e-006 6.55923e-006 7.74547e-006 1.90374e-006 4.59696e-006 1.40214e-005 2.13445e-006 ...
2.95458e-006 1.40313e-006 4.35322e-006 2.97236e-006 2.54087e-006 3.9007e-006 1.83732e-006 6.17049e-006 4.5706e-006 ...
0.000308139 6.16358e-005 2.94314e-006 0.000366169 2.15676e-006 6.94888e-006 9.12826e-008 1.34615e-007 2.59118e-006 ...
1.56666e-007 1.87825e-006 3.41399e-011 3.018e-006 1.08268e-006 7.3329e-006 1.82816e-006 2.16708e-006 7.1705e-006 ...
1.16484e-006 3.08494e-006 5.49772e-006 8.56417e-007 4.1841e-006 1.34978e-005 3.75113e-006 7.56571e-006 1.27116e-005 ...
7.40734e-006 6.04575e-007 8.15565e-006 7.99693e-006 5.98616e-005 6.84026e-006 8.56417e-007 4.1841e-006 1.34978e-005 ...
3.75113e-006 7.56571e-006 1.27116e-005 4.68827e-006 9.47876e-006 1.02502e-005 1.81397e-006 7.68073e-006 1.03926e-005];
err_qmoy50=[...
4.2092e-006 3.2285e-006 4.84967e-006 4.44505e-006 2.19084e-006 4.24879e-006 3.83829e-006 9.83341e-007 4.02424e-006 ...
7.95859e-006 0.00144335 3.66332e-006 5.39358e-006 1.27716e-005 1.54294e-006 3.5972e-006 4.82017e-006 2.16353e-006 ...
7.1988e-006 3.7204e-006 5.02521e-006 6.79124e-007 3.03677e-006 5.84706e-006 5.55324e-006 5.15189e-005 0.00345197 ...
9.68365e-006 1.39887e-006 2.469e-006 8.99713e-006 7.77037e-006 1.42496e-005 3.45773e-006 4.91636e-006 4.12828e-006 ...
2.57292e-006 1.99015e-006 6.43187e-006 5.46157e-006 3.00613e-006 1.18787e-006 4.23845e-006 1.02397e-005 1.64181e-011 ...
2.94742e-006 1.87914e-006 3.07712e-006 2.44443e-006 2.51125e-006 2.93587e-006 2.90492e-007 4.68125e-006 2.68398e-006 ...
4.73007e-005 6.16959e-005 9.60124e-006 0.000365646 6.63415e-007 5.1332e-006 3.82406e-007 1.15759e-007 4.15626e-006 ...
1.93411e-007 9.79063e-007 4.66069e-006 1.86997e-006 1.05606e-006 8.34544e-006 7.75831e-007 1.70587e-006 5.46603e-006 ...
1.19593e-006 1.99117e-006 5.08331e-006 2.04431e-006 1.73228e-011 1.01317e-005 3.03081e-006 3.85662e-006 9.58567e-006 ...
5.21956e-006 3.15269e-007 7.15812e-006 3.77947e-006 7.44978e-005 6.17868e-006 2.04431e-006 1.73228e-011 1.01317e-005 ...
3.03081e-006 3.85662e-006 9.58567e-006 4.65796e-006 4.32777e-006 8.96968e-006 2.89539e-006 1.01126e-005 9.60765e-006];
err_qmoy40=[...
3.33632e-006 9.67021e-007 4.86951e-006 1.0576e-005 6.33417e-007 3.82393e-006 2.85492e-006 2.27147e-009 2.16781e-006 ...
8.3243e-006 0.00149882 1.18463e-006 3.64129e-006 2.36695e-006 2.54859e-006 1.65546e-006 4.14841e-006 1.82504e-006 ...
1.87296e-006 1.38157e-006 3.84619e-006 2.94448e-006 3.03625e-006 5.02206e-006 5.60478e-006 5.13474e-005 7.32605e-005 ...
5.05535e-006 1.90523e-006 5.62124e-009 8.10528e-006 6.37415e-006 3.22389e-006 9.63728e-007 5.51491e-006 4.0848e-006 ...
2.2879e-006 1.33658e-006 5.34634e-006 4.602e-006 2.38582e-006 5.38741e-006 3.20416e-006 5.16037e-006 3.92046e-006 ...
3.43038e-006 1.8112e-006 2.35297e-010 1.64451e-006 2.12476e-006 1.92209e-006 1.40958e-007 1.67721e-006 1.92833e-006 ...
6.03399e-005 6.15035e-005 5.76726e-006 0.000337673 6.00345e-007 5.75779e-006 3.18937e-008 8.48636e-008 4.10802e-006 ...
1.1319e-007 9.33423e-007 5.68121e-011 1.6848e-006 1.67579e-006 7.37359e-008 6.40628e-007 6.16041e-006 5.36944e-006 ...
3.04278e-006 7.08433e-006 1.15845e-006 2.77824e-006 1.89517e-006 4.73452e-006 4.11589e-006 4.2544e-006 8.11276e-006 ...
5.2043e-006 1.25478e-007 5.24457e-006 2.7322e-006 6.80141e-005 6.05134e-006 2.77824e-006 1.89517e-006 4.73452e-006 ...
4.11589e-006 4.2544e-006 8.11276e-006 4.50621e-006 4.23288e-006 9.81289e-006 1.39734e-006 4.00879e-011 8.08088e-006];
err_qmoy30=[...
1.17842e-006 4.97155e-006 5.53108e-006 8.45764e-007 8.75219e-007 3.23149e-006 5.0679e-006 4.68897e-011 2.09738e-006 ...
7.51985e-006 0.00153574 8.06991e-007 4.33281e-006 3.08648e-006 2.56939e-006 1.60377e-006 2.0893e-006 3.93332e-006 ...
3.69946e-006 1.37703e-006 2.75269e-007 2.92772e-006 5.88974e-007 2.1809e-006 4.69178e-011 4.19522e-005 2.44153e-005 ...
6.67108e-006 3.54894e-007 5.62206e-009 8.14606e-006 6.05781e-006 2.48289e-006 9.39663e-007 4.33547e-006 3.87759e-006 ...
2.25751e-006 1.51915e-006 4.88476e-006 5.72858e-006 4.68167e-006 1.3413e-009 3.13389e-006 5.15012e-006 2.22374e-011 ...
3.68563e-006 9.8535e-007 2.10787e-010 3.50637e-006 2.08556e-006 2.25735e-007 5.55009e-007 2.4256e-007 3.61365e-007 ...
9.28239e-005 6.14483e-005 4.16338e-011 0.000292006 4.7302e-007 6.1635e-006 1.81816e-008 8.87066e-008 2.31762e-006 ...

6.40409e-008 6.43832e-006 1.82275e-006 1.0989e-006 6.87966e-007 5.64855e-009 4.57254e-007 4.46993e-006 1.69468e-006 ...
3.10117e-006 1.12191e-008 2.89972e-006 2.77273e-007 1.49935e-011 3.14144e-006 2.78813e-006 3.58339e-007 7.27013e-006 ...
2.49149e-006 5.72842e-008 3.90635e-006 5.01078e-006 6.63843e-005 3.04767e-006 2.77273e-007 1.49935e-011 3.14144e-006 ...
2.78813e-006 3.58339e-007 7.27013e-006 4.32881e-006 4.06888e-006 8.06495e-006 1.38287e-006 8.68877e-007 6.72731e-006];
err_qmoy20=[...
2.33865e-006 8.19277e-007 4.64382e-006 5.66737e-008 8.63789e-007 2.40728e-006 1.81349e-006 2.27435e-009 2.09842e-006 ...
2.8168e-006 3.7571e-005 4.99851e-006 2.70829e-007 4.41449e-011 5.13371e-007 1.10985e-006 6.26845e-007 4.08194e-006 ...
1.02086e-007 3.70645e-007 7.01048e-008 8.47908e-011 3.02962e-006 4.681e-006 1.33607e-007 6.70627e-005 1.49371e-005 ...
4.44266e-006 3.19733e-007 5.62224e-009 8.21993e-006 6.49093e-006 1.4322e-006 5.05711e-007 4.93201e-006 3.23297e-006 ...
1.55822e-006 6.7172e-007 5.39545e-006 4.02706e-006 2.55934e-012 3.9231e-011 1.52873e-006 1.19903e-007 1.00891e-006 ...
2.80315e-006 6.02214e-007 1.82518e-010 3.86348e-007 2.08231e-006 2.53338e-007 1.00486e-006 1.08757e-010 1.29507e-007 ...
1.955e-005 6.1576e-005 4.07118e-011 0.000281204 4.39676e-007 1.22898e-006 2.29568e-007 3.85862e-008 3.70485e-011 ...
1.52837e-007 3.26881e-006 7.41502e-011 3.02721e-008 4.74704e-007 1.96718e-006 3.4104e-007 6.37512e-008 4.51772e-007 ...
5.3543e-007 7.21963e-007 1.254e-006 1.11191e-007 1.28463e-011 2.80678e-006 2.74881e-006 4.21506e-011 7.1611e-006 ...
1.1618e-006 2.64982e-010 3.76418e-006 3.44508e-006 6.26782e-005 3.20117e-006 1.11191e-007 1.28463e-011 2.80678e-006 ...
2.74881e-006 4.21506e-011 7.1611e-006 4.80297e-006 3.81216e-011 8.13685e-006 2.05219e-006 4.10505e-011 6.68349e-006];
err_qmoy10=[...
3.5626e-008 4.08969e-011 2.25487e-006 7.28469e-007 2.7627e-011 1.79157e-006 5.21733e-012 4.19593e-011 1.92492e-006 ...
7.98193e-007 7.77042e-009 6.31581e-007 2.46631e-006 4.58359e-011 1.55954e-006 5.29579e-007 3.2322e-007 1.31269e-006 ...
2.94128e-006 6.282e-006 5.30405e-008 8.88926e-011 3.27817e-007 2.06648e-006 5.06435e-011 2.33303e-005 2.04517e-010 ...
2.43276e-006 2.36081e-007 4.45008e-011 8.29874e-006 2.06582e-006 8.88589e-011 4.17067e-007 4.16949e-011 6.6101e-007 ...
1.64619e-007 3.04954e-011 1.02967e-006 3.35033e-012 3.21537e-012 3.99609e-011 8.46632e-007 6.60249e-007 1.25879e-007 ...
6.78561e-006 8.90283e-009 4.79326e-011 2.43333e-009 1.96224e-007 4.95124e-011 3.51711e-009 0 2.73463e-011 ...
1.59056e-005 5.78929e-005 3.86045e-011 1.65855e-012 4.17964e-007 4.85719e-011 2.32369e-009 0 2.92291e-006 ...
1.25148e-008 1.03808e-006 8.14514e-011 8.87971e-008 3.6135e-007 4.50858e-011 3.47531e-007 8.69908e-009 8.41642e-007 ...
2.88887e-007 4.56231e-011 3.61548e-012 5.60521e-008 1.2702e-011 2.80904e-006 2.50339e-006 3.24371e-006 7.21049e-006 ...
6.2959e-007 2.71045e-010 2.08183e-006 4.00988e-006 6.01337e-005 6.69778e-007 5.60521e-008 1.2702e-011 2.80904e-006 ...
2.50339e-006 3.24371e-006 7.21049e-006 4.54506e-006 7.45555e-007 8.85084e-006 1.36623e-006 4.03063e-011 5.80139e-006];

```
% calcul de l'intervalle de confiance selon la distribution de Student
err_QMoy_Q=[Stu_Const*std(err_qmoy95)/sqrt(length(err_qmoy95))...
            Stu_Const*std(err_qmoy90)/sqrt(length(err_qmoy90))...
            Stu_Const*std(err_qmoy80)/sqrt(length(err_qmoy80))...
            Stu_Const*std(err_qmoy70)/sqrt(length(err_qmoy70))...
            Stu_Const*std(err_qmoy60)/sqrt(length(err_qmoy60))...
            Stu_Const*std(err_qmoy50)/sqrt(length(err_qmoy50))...
            Stu_Const*std(err_qmoy40)/sqrt(length(err_qmoy40))...
            Stu_Const*std(err_qmoy30)/sqrt(length(err_qmoy30))...
            Stu_Const*std(err_qmoy20)/sqrt(length(err_qmoy20))...
            Stu_Const*std(err_qmoy10)/sqrt(length(err_qmoy10))];
% calcul de moyenne
moy_QMoy_Q=[mean(err_qmoy95) mean(err_qmoy90) mean(err_qmoy80)...
mean(err_qmoy70) mean(err_qmoy60)  mean(err_qmoy50) mean(err_qmoy40)...
mean(err_qmoy30) mean(err_qmoy20) mean(err_qmoy10) ];%calcul des moyennes
```

## B.2.3 Listing du programme « devG.m » contenant les données des comparaisons numérique.

```
err_max95=[...
0.000995278 0.000974691 0.000654511 0.000995278 0.000863532 0.000384808 0.00104265 0.00123032 0.00160941 ...
0.00209807 0.000750209 0.000691201 0.00110384 0.000707194 0.00128246 0.00108526 0.00046006 0.00144706 ...
0.000971674 0.00122744 0.00202898 0.00010714 0.000677439 0.00137438 0.000909943 0.00154213 0.000696857 ...
0.000791783 0.00494668 0.00161559 0.00113985 0.00151554 0.000265443 0.00111453 0.00141414 0.000603817 ...
0.00187615 0.0011025 0.00156641 0.00113843 0.00190299 0.00129759 0.00089521 0.000565113 0.0019716 ...
0.000692252 0.00166747 0.0008019 0.00110408 0.00135907 5.76747e-005 0.00145841 0.000610662 0.00120987 ...
0.000759105 0.000572621 0.00120995 0.000377455 0.00145596 0.000971366 0.0011528 0.00131156 0.000499588 ...
0.000999092 0.00110051 0.00146295 0.00120566 0.00015186 0.00128823 0.00194236 0.000752195 0.0022158 ...
0.00122904 0.000641119 0.00266714 0.000150674 0.000478282 0.000313549 0.000220295 0.000636764 0.000222562 ...
0.00080707 0.00101949 0.00119786 0.000961816 0.000115319 0.00137485 0.000880766 0.00448157 0.00149686 ...
0.000947774 7.06222e-006 0.000902177 0.00080707 0.00101949 0.00119786 0.000961816 0.000115319 0.00137485];
err_max90=[...
0.00020698 0.00212453 0.000263727 0.000206701 0.000857634 0.00031315 0.000473819 0.00105615 0.000284574 ...
0.000562743 0.000743984 0.000302092 0.000547142 0.00137182 0.000816048 0.000596883 0.000127316 0.000693841 ...
0.000431149 0.00131673 0.000427331 0.000243345 0.000510722 0.000486071 0.000570603 0.000375353 0.000508192 ...
0.0012682 0.0025867 0.000617517 0.00133289 0.000519128 0.000265443 0.000680468 0.000552453 0.00202718 ...
0.00166781 0.000744147 0.000585479 0.000798759 0.00140784 0.000439649 0.000370218 0.000886321 0.000505511 ...
0.000451628 0.000504612 0.00140633 0.000416446 0.000525506 0.00052722 0.000682308 0.000162994 0.00118426 ...
0.00019639 0.000313122 0.000192678 0.000176504 0.000405386 0.00100496 0.00031854 0.000429264 0.000356541 ...
0.00128343 0.000375079 0.00113942 0.000649224 0.000326355 0.000335015 0.000638447 0.00138774 0.000645493 ...
0.000401711 0.000654386 0.00178488 1.86141e-005 0.00176788 0.000313549 2.70702e-005 0.00117197 0.000269093 ...
0.000536621 0.000884633 0.000583538 0.000445801 0.000270515 0.000789905 0.000643418 0.00400834 0.00044364 ...
0.000356313 7.06222e-006 0.000707101 0.000536621 0.000884633 0.000583538 0.000445801 0.000270515 0.000789905];
err_max80=[...
4.9858e-005 0.00100291 0.000218164 0.000153141 0.000802481 0.00031315 0.000271153 0.000811799 0.000270608 ...
0.000100229 0.000743984 0.000380551 0.000218151 0.000816146 0.000403133 0.000145808 0.000371592 0.000542126 ...
0.000244938 0.000680193 0.000241383 3.38976e-005 0.00139454 0.000279548 0.000995207 0.000317015 9.14775e-005 ...
0.000998763 0.000621252 0.000237752 0.000919937 0.000311305 0.000265443 0.000693617 0.000336939 0.000749667 ...
0.000277958 0.000237752 0.000575523 0.000425715 0.000811605 0.000197556 0.000460801 0.000462779 0.000162699 ...
0.000149024 0.000262561 0.00081594 0.000204648 0.000142444 0.000146347 0.000387028 7.51716e-005 0.00114317 ...
0.000111035 0.000152614 0.000262487 0.0001463 0.000384878 0.000987209 0.00013413 0.000190419 0.000316569 ...
0.000819694 0.000140712 0.000785802 0.000496661 0.00015186 0.00013886 0.000638447 0.000638465 0.000210036 ...
0.000157576 0.001171 0.000448848 8.17952e-006 0.00108314 2.41084e-005 4.20799e-006 0.000911397 0.000118227 ...
0.000457353 0.000887985 0.000186091 0.000287448 0.000215408 0.000233882 0.000180047 0.00649142 0.00044364 ...
0.000158003 7.06222e-006 0.000707101 0.000457353 0.000887985 0.000186091 0.000287448 0.000215408 0.000233882];
err_max70=[...
4.2386e-005 0.00133182 0.000218164 9.94073e-005 0.000383908 0.000188959 6.392e-005 0.00123615 0.000249723 ...
5.82467e-005 0.000439359 0.000739251 0.000218151 0.00132569 0.000278435 8.10079e-005 0.000617288 0.000542126 ...
0.000244938 0.000241743 0.000123784 2.87569e-005 0.000125569 7.70633e-005 0.000995207 0.000264952 9.00906e-005 ...
0.000631361 0.000621252 0.000191278 0.000933879 0.000364663 0.000256143 0.000456797 0.000764251 0.000746816 ...
5.52814e-005 0.000107796 0.00054403 0.000354354 0.000600481 0.000383031 0.000241542 0.000470269 0.000457791 ...
6.93354e-005 0.000262561 0.00081594 0.000509783 6.73736e-005 0.000272067 0.000239124 7.09478e-005 0.000895257 ...
0.000161708 0.000341464 0.000195467 0.0001463 4.34459e-005 0.00118784 0.00024213 5.42466e-005 0.000139515 ...
0.000403298 5.27306e-005 0.00187967 0.000496661 2.31922e-005 0.000683793 0.000453187 0.000555819 4.39502e-005 ...
0.000157576 0.000503824 0.000138852 8.17952e-006 0.000759632 2.76392e-005 4.20799e-006 0.00114452 0.000118227 ...
0.000227582 0.000287549 5.83624e-005 0.000180011 0.000215408 0.00050682 0.000277261 0.0046869 0.000886274 ...
0.000158003 7.06222e-006 0.000707101 0.000227582 0.000287549 5.83624e-005 0.000180011 0.000215408 0.00050682];
err_max60=[...
2.28794e-005 0.00133182 1.43002e-005 1.83856e-005 0.000417953 0.00031315 0.000271153 0.000371591 0.000108698 ...
6.53377e-005 0.000439359 0.000148562 0.000218151 0.000900171 0.000274114 0.000335469 0.00031852 0.000857957 ...
0.000246904 0.000392162 0.000123784 3.38976e-005 1.14063e-009 0.000675947 0.000747402 0.000120404 9.00906e-005 ...
0.000605977 0.000232594 7.2197e-005 0.00039393 0.000145194 8.88616e-005 0.000450324 0.000244284 0.000790397 ...
5.52814e-005 4.89518e-005 0.000457468 0.000295503 1.00965e-006 0.00035822 0.000460801 0.000470269 0.000180743 ...
0.000284075 0.000262561 9.84296e-007 0.000580574 4.25292e-005 0.000117694 0.000717075 7.09478e-005 0.00114317 ...
0.000182427 0.000341464 0.000224621 0.0001463 4.34459e-005 3.18491e-007 0.000470589 0.000190419 0.000139515 ...
0.000218466 6.32021e-005 0.00114553 8.40866e-005 0.000326355 0.000284323 0.000453909 0.000745657 3.3548e-005 ...
0.000157576 0.000503824 0.000438614 6.05596e-006 0.000563133 0.000393114 0.000210847 0.00114452 5.6385e-006 ...
7.72605e-005 1.35532e-006 5.31244e-005 0.000180011 0.00010134 0.000171027 0.000277261 0.00682977 0.00044364 ...
0.000158003 7.06222e-006 0.000707101 7.72605e-005 1.35532e-006 5.31244e-005 0.000180011 0.00010134 0.000171027];
```

```
err_max50=[...
0.000126325 0.00133182 4.722e-005 1.29316e-005 0.000417953 8.979e-005 0.000271153 0.00107887 0.000273195 ...
9.80746e-005 0.000102152 0.000494397 0.000218151 7.80437e-007 0.000274114 0.000335469 0.0003735 0.000300902 ...
0.000244938 2.39909e-005 0.000123784 4.57005e-005 0.000385938 8.85858e-005 0.000570603 0.000264952 9.00906e-005 ...
0.000452127 0.000586876 4.83832e-005 0.00100209 0.000347244 0.000329595 0.000450324 0.000176017 1.01328e-006 ...
5.52814e-005 2.40763e-005 0.000455924 0.000190869 2.227e-009 0.000177524 0.000460801 0.000191871 0.000418746 ...
0.000149024 0.000262561 1.01473e-006 0.000580574 2.41653e-005 0.00014555 0.000154883 1.4796e-005 0.000765294 ...
0.000182427 0.000341464 0.00016911 8.19329e-005 4.34459e-005 5.08944e-007 0.000146925 1.04979e-005 0.000139515 ...
0.00030865 0.000140909 0.00114553 9.75107e-005 0.000164963 0.000683793 0.0009746 0.000250218 3.43801e-005 ...
0.000157576 0.000582119 0.000705945 6.05596e-006 8.06549e-010 2.44738e-006 3.21582e-006 0.00138276 0.000250596 ...
3.69806e-005 2.87572e-008 0.000573042 0.000180011 3.3814e-005 5.51545e-005 0.000107483 0.00425599 0.000571679 ...
0.000158003 6.93227e-006 0.000341337 3.69806e-005 2.87572e-008 0.000573042 0.000180011 3.3814e-005 5.51545e-005];
err_max40=[...
5.63142e-006 1.36014e-006 5.05455e-006 9.23449e-006 0.000290683 3.06244e-005 2.24635e-006 0.00107887 0.00014468 ...
1.49757e-006 1.09026e-007 0.000568004 0.000218151 0.00132569 0.000147171 7.84748e-005 7.48859e-005 4.39144e-005 ...
7.72406e-007 2.39909e-005 0.000123784 4.57005e-005 0.000241824 1.69047e-005 0.000570603 7.06662e-006 2.00421e-006 ...
1.45095e-007 5.82359e-006 2.50091e-005 0.00100209 0.000189303 0.000265443 9.76825e-005 0.00018651 1.01328e-006 ...
0.000153292 2.73387e-005 0.000121065 0.00017718 2.05423e-009 0.000236582 8.00192e-006 0.000340177 3.00467e-005 ...
1.64701e-005 0.000262561 1.01473e-006 0.000580574 0.000116783 2.09951e-006 5.65335e-005 8.95987e-005 8.06549e-010 ...
0.000182427 7.04329e-006 6.73998e-005 0.00017875 1.01915e-006 5.08944e-007 4.63787e-005 1.02365e-006 1.56463e-006 ...
0.000216494 0.000140909 0.00114553 0.000123787 2.23601e-005 0.00013886 0.000389793 9.71366e-007 9.44872e-005 ...
0.000157576 3.14568e-006 0.000601943 0.000194157 4.43922e-007 2.38475e-006 3.04422e-006 0.00138276 0.000110017 ...
2.31644e-005 8.06549e-010 0.000573042 0.000180011 3.24294e-005 5.51545e-005 0.000521835 0.00487303 0.000244182 ...
2.32872e-005 6.93227e-006 0.000341337 2.31644e-005 8.06549e-010 0.000573042 0.000180011 3.24294e-005 5.51545e-005];
err_max30=[...
5.63142e-006 9.31323e-010 5.01543e-006 4.80759e-006 2.3221e-006 3.06244e-005 1.49757e-006 8.06549e-010 0.000111227 ...
1.49757e-006 4.64496e-009 0.000146685 0.000218151 8.06549e-010 2.50126e-005 8.01356e-005 6.58545e-010 0.000894262 ...
8.05038e-007 2.39909e-005 0.000123784 4.57005e-005 3.58366e-007 8.85858e-005 0.000570603 3.19404e-006 9.22915e-007 ...
1.97911e-006 1.04107e-005 7.15174e-006 3.49552e-007 0.000139542 0.000265443 2.79144e-006 0.000104118 1.01328e-006 ...
4.98221e-006 9.99371e-006 3.50219e-007 2.77117e-005 8.06549e-010 0.000101709 8.00192e-006 1.82177e-006 3.7762e-005 ...
1.62035e-005 0.000165544 8.06549e-010 1.91366e-005 1.09315e-006 2.09951e-006 0.000325261 1.89552e-006 8.06549e-010 ...
2.2284e-006 3.17965e-005 6.58545e-010 1.3759e-005 1.01915e-006 6.58545e-010 3.84288e-006 9.00016e-007 1.56123e-006 ...
0.00021583 2.12288e-006 0.000149389 9.4291e-005 2.23601e-005 3.65612e-006 0.000389793 9.07133e-007 9.44872e-005 ...
2.38573e-006 3.1751e-006 0.000322694 6.05596e-006 8.06549e-010 2.11614e-006 1.71885e-006 8.06549e-010 8.84651e-005 ...
1.63664e-005 8.06549e-010 4.16892e-005 0.000151868 2.32831e-010 1.64099e-005 0.000291168 0.00487303 0.000216173 ...
1.33405e-005 6.93227e-006 0.000399012 1.63664e-005 8.06549e-010 4.16892e-005 0.000151868 2.32831e-010 1.64099e-005];
err_max20=[...
1.74378e-007 9.31323e-010 5.01716e-006 6.91828e-005 2.54603e-006 2.70674e-006 9.72078e-007 9.31323e-010 1.99407e-006 ...
8.21942e-007 1.27708e-007 1.56812e-005 6.8145e-007 9.31323e-010 7.91491e-006 9.61988e-007 6.58545e-010 7.04591e-005 ...
1.03388e-006 2.39909e-005 0.000123784 4.57005e-005 5.49284e-007 1.2821e-005 0.000570603 3.19404e-006 9.22915e-007 ...
2.07411e-006 2.66936e-006 4.99187e-006 9.31323e-010 4.01664e-006 5.0051e-010 1.61232e-007 1.10921e-005 1.01328e-006 ...
3.84813e-005 4.03246e-006 1.93744e-007 2.30787e-005 8.06549e-010 8.49603e-006 4.64866e-006 1.82177e-006 1.71716e-005 ...
9.76239e-006 9.78991e-007 8.06549e-010 6.78837e-007 1.09315e-006 2.09951e-006 4.15864e-006 2.17576e-007 8.06549e-010 ...
1.27152e-018 2.22207e-007 6.58545e-010 4.99968e-007 1.01915e-006 8.06549e-010 2.31781e-006 8.57752e-007 3.07264e-007 ...
3.6033e-006 2.12288e-006 0.000149389 6.34959e-006 1.90689e-005 3.83142e-006 7.80794e-005 9.31323e-010 2.22022e-005 ...
7.17946e-007 1.00684e-006 2.23473e-005 5.67769e-006 6.58545e-010 1.54197e-006 1.71885e-006 0.000154723 8.84651e-005 ...
1.40018e-005 8.06549e-010 9.8307e-006 9.37686e-005 2.32831e-010 7.12398e-005 2.07132e-005 0.00487303 6.10577e-006 ...
1.84529e-005 6.93227e-006 1.92212e-006 1.40018e-005 8.06549e-010 9.8307e-006 9.37686e-005 2.32831e-010 7.12398e-005];
err_max10=[...
9.58645e-008 6.58545e-010 0 1.92157e-007 1.81141e-006 5.6424e-007 6.55536e-007 9.31323e-010 5.2481e-007 ...
7.0197e-007 3.29272e-010 1.42921e-006 8.70682e-007 9.31323e-010 6.50472e-008 9.61988e-007 6.58545e-010 9.5252e-005 ...
8.44955e-007 2.39909e-005 2.49683e-006 6.27219e-008 2.48719e-007 3.34454e-006 0.000570603 3.19404e-006 9.22915e-007 ...
1.14063e-009 2.3615e-006 8.58636e-005 7.36275e-010 1.25594e-006 5.29396e-023 1.61232e-007 1.10921e-005 1.01328e-006 ...
1.76493e-006 2.8251e-006 1.93744e-007 1.36177e-005 8.06549e-010 5.34547e-007 5.44666e-007 3.29272e-010 2.72274e-005 ...
6.10856e-007 9.45545e-007 8.06549e-010 7.83469e-007 7.92433e-007 3.29272e-010 1.11854e-006 2.27013e-007 8.06549e-010 ...
6.28736e-019 2.43576e-007 6.58545e-010 4.99968e-007 7.35298e-007 9.31323e-010 5.74795e-007 8.5475e-007 4.65661e-010 ...
1.73579e-006 2.12288e-006 0.000149389 8.40866e-005 0 3.83142e-006 6.00654e-006 9.31323e-010 4.24402e-006 ...
7.17946e-007 1.31709e-009 5.53462e-005 6.05596e-006 8.06549e-010 1.54197e-006 1.71885e-006 8.06549e-010 5.6385e-006 ...
1.15458e-006 8.06549e-010 1.48167e-006 1.32957e-006 2.32831e-010 8.35342e-007 1.13246e-006 0.00406092 6.10577e-006 ...
5.34803e-007 6.94418e-006 2.48315e-006 1.15458e-006 8.06549e-010 1.48167e-006 1.32957e-006 2.32831e-010 8.35342e-007];
```

```
err_Max_G=[Stu_Const*std(err_max95)/sqrt(length(err_max95))...
           Stu_Const*std(err_max90)/sqrt(length(err_max90))...
           Stu_Const*std(err_max80)/sqrt(length(err_max80))...
          Stu_Const*std(err_max70)/sqrt(length(err_max70))...
          Stu_Const*std(err_max60)/sqrt(length(err_max60))...
          Stu_Const*std(err_max50)/sqrt(length(err_max50))...
          Stu_Const*std(err_max40)/sqrt(length(err_max40))...
          Stu_Const*std(err_max30)/sqrt(length(err_max30))...
          Stu_Const*std(err_max20)/sqrt(length(err_max20))...
          Stu_Const*std(err_max10)/sqrt(length(err_max10))];
moy_Max_G=[mean(err_max95) mean(err_max90) mean(err_max80) mean(err_max70)...
mean(err_max60) mean(err_max50) mean(err_max40) mean(err_max30) mean(err_max20)...
mean(err_max10) ];%calcul des moyennes
```

```
err_moy95=[...
1.79957e-005 3.60524e-007 5.74554e-005 1.52385e-005 1.96207e-007 1.26048e-005 5.82634e-005 4.28862e-007 0.000289702 ...
7.5024e-005 2.23688e-007 6.96628e-005 0.000110794 2.87336e-007 8.66887e-005 9.92635e-005 9.43898e-008 0.000124416 ...
6.32564e-005 4.47762e-007 0.000203801 2.49471e-006 1.22544e-007 6.00735e-005 1.37237e-007 0.000158355 4.8113e-005 ...
9.7901e-008 0.000410515 0.000119625 1.63887e-007 0.000278261 6.56979e-008 1.62806e-007 0.000124051 1.30252e-007 ...
0.000188903 0.000121889 2.94556e-007 9.84113e-005 5.10708e-007 0.000132068 4.65546e-005 8.12786e-008 9.9581e-005 ...
5.61436e-005 0.000139036 1.88586e-007 0.000160939 0.000204526 8.43135e-009 0.000107754 3.44925e-005 4.42732e-007 ...
5.22969e-005 3.37594e-005 2.06422e-007 3.7385e-005 9.59334e-005 3.10037e-007 0.000107225 7.38456e-005 1.32633e-007 ...
1.81176e-005 9.66958e-005 2.09396e-007 0.000146229 3.23523e-006 2.87756e-007 8.72043e-005 4.57166e-007 0.000197594 ...
0.000102013 1.15408e-007 0.000125091 2.89064e-006 1.32087e-007 7.8733e-006 3.55629e-006 1.38112e-007 7.66593e-006 ...
0.000186981 2.98215e-007 0.000175968 0.000132419 3.18948e-008 0.00012068 5.07009e-005 2.53738e-006 0.000136793 ...
5.39556e-005 8.68212e-007 5.27598e-005 0.000186981 2.98215e-007 0.000175968 0.000132419 3.18948e-008 0.00012068];
err_moy90=[...
3.98946e-006 5.66142e-007 1.17619e-005 4.19607e-006 1.87984e-007 3.32748e-006 9.15449e-006 6.31476e-007 2.78742e-005 ...
9.05149e-006 2.49142e-007 1.97038e-005 2.53016e-005 4.05568e-007 3.65635e-005 2.4758e-005 5.1342e-008 6.32952e-005 ...
1.38489e-005 2.7506e-007 1.43158e-005 7.50507e-007 3.33492e-008 1.32367e-005 1.72392e-007 1.82251e-005 7.25233e-006 ...
2.35896e-007 0.000184606 2.6483e-005 5.26773e-007 3.8776e-005 1.23131e-007 8.03189e-008 2.2907e-005 6.05961e-007 ...
4.04916e-005 1.80688e-005 1.17178e-007 3.37863e-005 6.04377e-007 4.02449e-005 5.9953e-006 1.12547e-007 3.85197e-005 ...
1.73054e-005 2.45019e-005 3.75292e-007 4.66903e-005 3.10479e-005 1.38118e-007 3.3487e-005 3.33551e-006 3.36334e-007 ...
1.01562e-005 3.18725e-006 8.23385e-009 9.77298e-006 6.00883e-006 5.67819e-007 7.93807e-006 6.22332e-006 1.75972e-007 ...
1.9092e-006 1.03985e-005 4.89047e-007 2.51519e-005 1.99984e-006 4.00208e-008 1.4162e-005 3.97064e-007 3.7045e-005 ...
9.31652e-006 1.22477e-007 8.2271e-005 9.4477e-007 7.57853e-007 1.23931e-006 9.97937e-007 3.84978e-007 3.40867e-006 ...
6.15421e-005 4.18834e-007 7.04572e-005 3.07593e-005 1.77013e-007 2.72517e-005 1.674e-005 2.75245e-006 4.54349e-005 ...
1.48262e-005 1.32757e-006 1.90805e-005 6.15421e-005 4.18834e-007 7.04572e-005 3.07593e-005 1.77013e-007 2.72517e-005 ];
err_moy80=[...
5.97414e-007 9.67044e-007 1.37892e-006 6.26966e-007 1.5325e-007 1.60547e-006 1.64966e-006 1.02286e-006 3.74267e-006 ...
7.33187e-007 1.69376e-007 4.94029e-006 2.85791e-006 8.47362e-007 5.50542e-006 2.09955e-006 1.4068e-007 2.33573e-005 ...
3.0688e-007 2.33726e-007 1.06794e-006 3.74374e-007 1.62912e-007 1.80625e-006 6.48594e-007 1.58745e-006 6.08896e-007 ...
9.68598e-008 1.69493e-006 4.19835e-006 2.98819e-007 7.13701e-006 4.7962e-008 1.25482e-007 3.59259e-006 3.39398e-007 ...
7.19069e-006 2.09715e-006 2.20459e-007 9.33155e-006 1.01053e-006 6.56691e-006 1.30207e-006 2.12311e-007 8.62942e-006 ...
1.60358e-006 3.42889e-006 1.08223e-006 7.16384e-006 4.06385e-006 3.44445e-008 5.40387e-006 3.47213e-007 1.23536e-007 ...
9.13876e-007 4.77121e-007 4.94626e-008 7.11647e-007 3.56566e-007 1.10332e-006 1.10604e-006 7.06044e-007 9.84206e-008 ...
8.97684e-007 1.07313e-006 4.81671e-007 2.67324e-006 8.11712e-007 1.34211e-008 4.32178e-006 2.3194e-007 5.02677e-006 ...
9.31551e-007 2.75801e-007 5.73701e-006 5.32242e-007 8.04978e-007 3.42845e-007 4.98251e-007 4.75835e-007 1.12327e-006 ...
1.08067e-005 3.78843e-007 1.95839e-005 1.12963e-005 1.39277e-007 1.40442e-005 3.63756e-006 2.18167e-006 4.72841e-006 ...
3.30657e-006 9.08342e-007 6.70399e-006 1.08067e-005 3.78843e-007 1.95839e-005 1.12963e-005 1.39277e-007 1.40442e-005];
err_moy70=[...
2.64474e-007 7.90063e-007 1.23402e-006 4.27257e-007 8.5821e-008 1.29285e-006 5.37161e-007 6.59211e-007 1.26684e-006 ...
2.35172e-007 7.42383e-008 2.30231e-006 1.08093e-006 1.03782e-006 2.6556e-006 7.41884e-007 1.43619e-007 9.6828e-006 ...
1.32113e-007 5.34086e-008 5.26326e-007 1.81847e-007 1.3556e-008 1.46025e-006 6.63993e-007 5.78526e-007 1.23374e-007 ...
1.64998e-007 1.20964e-006 7.96124e-007 3.44828e-007 1.71022e-006 8.1951e-008 1.31102e-007 1.48504e-006 5.67697e-007 ...
1.35335e-006 5.77757e-007 2.47849e-007 2.1031e-006 4.42831e-007 2.40104e-006 7.93753e-007 1.57115e-007 3.87252e-006 ...
6.61671e-007 5.82655e-007 5.74207e-007 1.46718e-006 8.73852e-007 6.80214e-008 1.66843e-006 2.58154e-007 1.55113e-007 ...
3.20145e-007 3.09639e-007 7.97365e-008 4.11934e-007 7.38028e-008 5.85679e-007 5.02189e-007 1.19534e-007 2.44834e-008 ...
6.23359e-007 2.82216e-007 7.68235e-007 1.46727e-006 2.91225e-007 1.33384e-007 6.16121e-006 1.39314e-007 1.31161e-006 ...
```

2.42597e-007 1.95223e-007 3.23778e-006 5.17243e-007 5.24931e-007 3.06537e-007 4.93805e-007 3.23579e-007 1.06825e-006 ...
5.47152e-006 1.26965e-007 5.03255e-006 4.85826e-006 2.2907e-007 3.76527e-006 1.74848e-006 2.21788e-006 2.15981e-006 ...
1.54812e-006 1.6916e-006 4.98113e-006 5.47152e-006 1.26965e-007 5.03255e-006 4.85826e-006 2.2907e-007 3.76527e-006];
err_moy60=[...
9.96714e-008 3.0157e-007 4.62379e-007 1.67684e-007 1.21342e-007 7.31074e-007 1.295e-007 3.20065e-008 8.74238e-007 ...
1.55266e-007 8.19182e-008 1.10346e-006 2.55526e-007 7.81405e-008 1.14554e-006 3.46269e-007 7.94246e-008 5.57486e-006 ...
2.02966e-007 4.48417e-008 4.30855e-007 2.04164e-007 8.22918e-012 1.43081e-006 3.58788e-007 3.63336e-007 7.88826e-008 ...
1.57805e-007 5.50488e-007 5.90659e-007 5.57983e-008 8.19202e-007 2.57895e-008 8.36632e-008 1.27329e-006 1.70407e-007 ...
6.7908e-007 2.87143e-007 2.28799e-007 1.47303e-006 2.62553e-010 1.2815e-006 4.78757e-007 1.10388e-007 1.76436e-006 ...
4.75583e-007 3.42295e-007 2.51352e-010 5.65127e-007 2.90574e-007 1.77212e-008 1.14614e-006 2.03657e-007 1.31238e-007 ...
2.47338e-007 2.19006e-007 1.02557e-007 3.39847e-007 5.5987e-008 1.00365e-010 4.67142e-007 1.21284e-007 5.9972e-008 ...
5.42002e-007 1.86461e-007 2.44053e-007 1.05102e-006 2.04066e-007 2.64901e-008 2.5297e-006 6.23864e-008 1.17436e-006 ...
1.35023e-007 1.35114e-007 1.23778e-006 1.52374e-007 7.17347e-008 3.25644e-007 2.72466e-007 5.355e-007 3.46798e-007 ...
2.27724e-006 1.08988e-010 1.54613e-006 4.25865e-006 1.19902e-007 1.75479e-006 1.42807e-006 3.00381e-006 1.03539e-006 ...
1.33992e-006 2.16539e-006 3.77427e-006 2.27724e-006 1.08988e-010 1.54613e-006 4.25865e-006 1.19902e-007 1.75479e-006 ];
err_moy50=[...
1.03396e-007 1.06844e-007 3.03045e-007 7.26802e-008 1.07325e-007 4.45053e-007 8.9604e-008 1.32412e-007 6.69315e-007 ...
5.55441e-008 2.05967e-008 1.03135e-006 1.1404e-007 1.29156e-010 8.69609e-007 1.5316e-007 1.15141e-007 2.66792e-006 ...
1.2958e-007 1.79599e-008 2.99767e-007 1.87134e-007 1.11223e-007 1.19024e-006 8.98492e-008 3.43e-007 1.18456e-007 ...
3.52128e-008 5.22861e-007 4.37758e-007 2.34894e-007 5.80718e-007 4.63627e-008 9.65244e-008 6.622e-007 1.08777e-009 ...
5.01646e-007 2.43135e-007 7.29141e-008 5.29103e-007 1.12871e-011 1.10134e-006 2.53826e-007 4.68308e-008 1.1783e-006 ...
1.75992e-007 1.46256e-007 2.61094e-010 4.63299e-007 1.3474e-007 2.0978e-008 6.46161e-007 1.05646e-007 3.94738e-008 ...
2.17014e-007 1.69808e-007 5.78832e-008 2.71533e-007 3.09463e-008 1.24136e-010 3.04795e-007 2.41788e-008 4.15198e-008 ...
4.28282e-007 9.63309e-008 1.30167e-007 8.17713e-007 1.63785e-007 4.83577e-008 1.69471e-006 1.79351e-008 6.85523e-007 ...
9.46839e-008 1.1123e-007 8.38037e-007 9.729e-008 9.51864e-012 1.25088e-007 1.27271e-007 5.79192e-007 3.86914e-007 ...
1.44671e-006 1.27951e-011 1.16761e-006 1.90768e-006 4.88663e-008 1.15763e-006 1.85852e-006 1.84568e-006 7.35079e-007 ...
1.28988e-006 8.95313e-007 2.43107e-006 1.44671e-006 1.27951e-011 1.16761e-006 1.90768e-006 4.88663e-008 1.15763e-006 ];
err_moy40=[...
2.23039e-008 7.66597e-011 2.32528e-007 3.15581e-008 1.77044e-008 3.93277e-007 1.25118e-008 1.04908e-007 3.19629e-007 ...
9.93015e-009 1.52806e-011 9.26637e-007 1.00359e-007 7.94324e-008 6.17601e-007 5.90272e-008 1.13423e-008 1.53833e-006 ...
6.77796e-009 1.79466e-008 2.02473e-007 1.885e-007 6.14499e-008 5.06934e-007 6.73723e-008 1.66026e-007 9.11781e-009 ...
9.4778e-011 3.04307e-007 3.29924e-007 1.25865e-007 3.46361e-007 9.77227e-008 1.34256e-008 5.51217e-007 9.77641e-010 ...
3.26849e-007 2.40909e-007 1.7293e-008 4.3888e-007 9.65245e-012 7.07849e-007 8.2993e-008 5.1295e-008 7.39461e-007 ...
9.48883e-008 6.95208e-008 2.56717e-010 2.78439e-007 4.75382e-008 1.51685e-010 4.64872e-007 5.93957e-008 1.2066e-011 ...
1.93821e-007 3.48992e-008 1.16565e-008 1.8551e-007 1.14427e-008 9.65791e-011 1.70499e-007 1.22102e-008 3.03479e-010 ...
2.7287e-007 4.10981e-008 1.30193e-007 6.29562e-007 1.46049e-007 8.04699e-009 9.45555e-007 6.25445e-010 4.28525e-007 ...
5.13887e-008 4.87636e-010 7.50854e-007 1.28644e-007 2.98914e-011 9.38578e-008 8.24785e-008 2.98079e-007 2.41853e-007 ...
9.11677e-007 9.85248e-012 8.9028e-007 1.14798e-006 4.68008e-009 8.94356e-007 1.44286e-006 1.30335e-006 5.11646e-007 ...
8.42201e-007 7.69019e-007 2.35456e-006 9.11677e-007 9.85248e-012 8.9028e-007 1.14798e-006 4.68008e-009 8.94356e-007];
err_moy30=[...
1.64781e-008 7.69155e-012 1.30043e-007 1.8833e-008 6.36396e-010 1.43691e-007 1.02694e-008 8.25202e-012 8.93847e-008 ...
6.63534e-009 8.03204e-012 3.46829e-007 1.43284e-007 8.47023e-012 2.35816e-007 2.57677e-008 2.97875e-012 1.17519e-006 ...
7.12821e-009 1.78619e-008 8.96801e-008 1.94018e-007 7.12402e-011 1.58561e-007 6.6594e-008 7.31013e-008 6.91905e-009 ...
2.14892e-010 1.73683e-007 2.04714e-007 4.7705e-011 1.73806e-007 5.1935e-008 3.28625e-010 2.77366e-007 7.26467e-010 ...
1.45291e-007 1.45583e-007 2.18721e-010 2.35043e-007 8.47663e-012 1.74391e-007 9.15387e-008 1.22879e-009 2.32518e-007 ...
8.1057e-008 3.27868e-008 8.07789e-012 7.05078e-008 2.03021e-008 1.50831e-010 2.09797e-007 2.39453e-008 1.21213e-011 ...
4.04633e-008 2.4437e-008 1.4195e-011 1.1799e-007 1.16573e-008 7.86492e-012 7.819e-008 1.19521e-008 1.67229e-010 ...
1.64099e-007 1.30614e-009 2.92426e-008 2.84528e-007 1.48856e-007 8.6864e-010 4.66782e-007 7.43693e-011 2.63568e-007 ...
1.04564e-009 5.57212e-010 4.13163e-007 5.42985e-008 7.58385e-012 2.3649e-008 4.03331e-008 1.69597e-011 1.59594e-007 ...
2.10867e-007 8.79724e-012 2.55897e-007 3.12271e-007 2.70889e-012 3.23682e-007 2.90807e-007 1.59686e-006 2.30662e-007 ...
1.03258e-007 4.73004e-007 2.03227e-006 2.10867e-007 8.79724e-012 2.55897e-007 3.12271e-007 2.70889e-012 3.23682e-007];
err_moy20=[...
3.81964e-009 8.54084e-012 1.7925e-008 1.25501e-008 7.58422e-010 7.04978e-009 9.66107e-009 8.63687e-012 2.82875e-009 ...
5.93014e-009 1.54573e-011 4.60226e-009 3.25193e-009 8.68572e-012 6.39992e-008 3.77173e-009 3.26177e-012 3.56196e-007 ...
7.12166e-009 1.7862e-008 2.23097e-008 1.66538e-007 1.21148e-010 2.64259e-008 6.65944e-008 8.36347e-009 8.11466e-009 ...
2.28724e-010 3.35952e-008 2.74149e-008 1.25952e-011 3.19529e-008 1.17058e-012 5.94588e-011 4.40496e-008 7.25877e-010 ...
2.87594e-008 2.36093e-008 7.15797e-011 5.43178e-008 8.70989e-012 1.86389e-008 2.86592e-008 5.95029e-010 6.12341e-008 ...
4.31011e-008 1.82506e-008 7.72869e-012 8.87323e-010 2.01189e-008 1.50313e-010 7.13789e-009 8.94368e-010 1.14234e-011 ...
2.82715e-021 1.01735e-009 1.37115e-011 1.22727e-010 1.13348e-008 7.95557e-012 9.8633e-009 1.12421e-008 5.13568e-011 ...
2.14908e-008 1.88203e-009 2.92426e-008 8.54615e-008 1.13379e-007 6.26891e-010 1.52663e-007 1.11879e-011 1.04859e-007 ...
6.16642e-010 3.35722e-010 1.26363e-007 4.90704e-008 7.63592e-012 1.14919e-008 3.60391e-008 1.43756e-008 1.44266e-007 ...
1.01001e-007 9.00003e-012 1.69714e-008 7.53057e-008 2.82124e-012 1.49953e-008 9.9713e-008 1.41724e-006 2.65008e-009 ...
6.75717e-008 4.73004e-007 1.12487e-008 1.01001e-007 9.00003e-012 1.69714e-008 7.53057e-008 2.82124e-012 1.49953e-008];
err_moy10=[...
5.81582e-011 7.94958e-012 0 2.72354e-010 1.1748e-010 2.14764e-010 2.67985e-009 8.11059e-012 1.70035e-010 ...
1.9573e-009 5.65462e-012 1.1324e-009 2.91898e-009 8.02566e-012 2.20003e-011 3.76566e-009 2.46384e-012 3.83607e-008 ...

2.34781e-009 1.78624e-008 8.01393e-009 6.85613e-011 9.5502e-011 6.19435e-009 6.65948e-008 6.35303e-009 2.77279e-009 ...
3.20608e-011 5.11839e-009 3.05389e-008 1.30961e-011 1.02741e-009 2.98779e-028 3.61162e-011 1.19266e-008 7.25992e-010 ...
1.11677e-009 1.04e-008 8.27853e-011 1.5953e-008 8.08347e-012 2.16384e-010 2.13203e-009 7.16125e-012 6.64638e-009 ...
1.83143e-009 7.38648e-009 7.9144e-012 6.67894e-010 8.35187e-009 4.11554e-012 3.25992e-009 2.36656e-010 1.1695e-011 ...
2.32772e-021 3.37344e-010 1.21285e-011 1.13908e-010 2.07358e-009 7.75074e-012 1.66411e-010 3.09302e-009 6.1156e-012 ...
8.92979e-010 8.82194e-010 2.92421e-008 1.84535e-008 0 3.75404e-010 1.03733e-008 1.06805e-011 6.97347e-009 ...
5.01035e-010 2.81854e-011 1.77007e-008 3.1605e-008 7.47689e-012 1.13295e-008 1.93283e-008 1.27328e-011 5.36698e-008 ...
5.84655e-009 8.63951e-012 2.07231e-010 5.69435e-009 1.45844e-012 1.00977e-009 1.95012e-009 1.5226e-006 2.05232e-009 ...
1.47502e-009 3.75159e-007 5.26172e-009 5.84655e-009 8.63951e-012 2.07231e-010 5.69435e-009 1.45844e-012 1.00977e-009 ];

% calcul de l'intervalle de confiance selon la distribution de Student
err_Moy_G=[Stu_Const*std(err_moy95)/sqrt(length(err_moy95))...
Stu_Const*std(err_moy90)/sqrt(length(err_moy90))...
Stu_Const*std(err_moy80)/sqrt(length(err_moy80))...
Stu_Const*std(err_moy70)/sqrt(length(err_moy70))...
Stu_Const*std(err_moy60)/sqrt(length(err_moy60))...
Stu_Const*std(err_moy50)/sqrt(length(err_moy50))...
Stu_Const*std(err_moy40)/sqrt(length(err_moy40))...
Stu_Const*std(err_moy30)/sqrt(length(err_moy30))...
Stu_Const*std(err_moy20)/sqrt(length(err_moy20))...
Stu_Const*std(err_moy10)/sqrt(length(err_moy10))];
moy_Moy_G=[mean(err_moy95) mean(err_moy90) mean(err_moy80) mean(err_moy70)...
mean(err_moy60) mean(err_moy50) mean(err_moy40) mean(err_moy30) mean(err_moy20)...
mean(err_moy10) ];%calcul des moyennes

err_qmoy95=[...
4.79707e-005 9.25276e-006 0.000108121 3.85447e-005 7.30734e-006 3.36073e-005 0.000129023 1.37076e-005 0.000504685 ...
0.000154695 7.35509e-006 0.000126651 0.000200335 7.40539e-006 0.000189445 0.000197866 3.63183e-006 0.000224683 ...
0.000131482 1.32938e-005 0.000400501 6.80096e-006 5.65966e-006 0.000126248 6.48298e-006 0.000289885 0.000101223 ...
4.86599e-006 0.000972404 0.000201971 7.49842e-006 0.000457793 2.18836e-006 8.17279e-006 0.000272612 5.1351e-006 ...
0.000333953 0.000209027 1.10893e-005 0.000212281 1.62418e-005 0.000244789 0.000105376 4.32017e-006 0.00020894 ...
0.000131312 0.000240723 6.89444e-006 0.000261023 0.000334517 4.43169e-007 0.000197988 8.34964e-005 1.12898e-005 ...
0.000113136 7.25051e-005 8.71202e-006 5.89771e-005 0.000195606 8.16394e-006 0.000219896 0.000151176 4.68849e-006 ...
5.84786e-005 0.000189407 1.03158e-005 0.000269048 7.53564e-006 1.17778e-005 0.000238787 1.10926e-005 0.000385073 ...
0.000186084 4.90563e-006 0.000352345 7.89784e-006 4.81348e-006 2.32636e-005 8.47363e-006 5.34505e-006 1.63496e-005 ...
0.000278545 1.00076e-005 0.000257478 0.000210852 1.17394e-006 0.000192839 0.000113768 5.37344e-005 0.000232347 ...
0.000130837 1.4387e-006 0.000115736 0.000278545 1.00076e-005 0.000257478 0.000210852 1.17394e-006 0.000192839 ];
err_qmoy90=[...
1.71618e-005 1.65401e-005 3.588e-005 1.71309e-005 6.89028e-006 1.19298e-005 4.09066e-005 1.24527e-005 6.02707e-005 ...
3.75845e-005 7.78093e-006 4.49499e-005 8.21917e-005 1.19373e-005 9.81468e-005 8.06425e-005 1.59397e-006 0.000120323 ...
5.32409e-005 1.13545e-005 4.56284e-005 2.77554e-006 2.9181e-006 4.50448e-005 5.96634e-006 5.59877e-005 2.85575e-005 ...
9.86561e-006 0.000491487 6.71785e-005 1.35544e-005 8.8164e-005 3.19948e-006 4.49918e-006 7.03677e-005 1.78308e-005 ...
9.57076e-005 6.0658e-005 4.82557e-006 0.00011363 1.42497e-005 8.49767e-005 2.49917e-005 6.40646e-006 8.08105e-005 ...
6.14453e-005 7.45784e-005 1.12103e-005 9.11487e-005 8.70598e-005 5.62584e-006 7.26213e-005 1.26466e-005 9.64318e-006 ...
3.22973e-005 1.39474e-005 8.00707e-007 2.10577e-005 2.65498e-005 1.3362e-005 2.60047e-005 2.66845e-005 4.82837e-006 ...
1.27054e-005 3.37746e-005 1.25787e-005 6.32148e-005 6.21552e-006 2.37882e-006 4.16868e-005 1.18146e-005 9.52971e-005 ...
3.16985e-005 5.47495e-006 0.000270792 1.62191e-006 2.09046e-005 4.81403e-006 1.80932e-006 1.00976e-005 5.44116e-006 ...
0.00012445 9.95848e-006 0.000114244 6.89651e-005 4.10148e-006 4.91604e-005 4.69442e-005 5.65244e-005 9.23369e-005 ...
4.76104e-005 2.09423e-006 4.56228e-005 0.00012445 9.95848e-006 0.000114244 6.89651e-005 4.10148e-006 4.91604e-005 ];
err_qmoy80=[...
3.21068e-006 1.45911e-005 5.50634e-006 4.0991e-006 5.75524e-006 6.35025e-006 8.12044e-006 1.50662e-005 9.00753e-006 ...
5.29139e-006 6.91193e-006 1.18162e-005 1.24535e-005 1.37644e-005 1.55672e-005 9.65384e-006 3.85346e-006 4.78822e-005 ...
3.74428e-006 6.75862e-006 4.28107e-006 1.24326e-006 9.14174e-006 5.82216e-006 1.34616e-005 5.67376e-006 4.42023e-006 ...
5.24105e-006 6.94177e-006 1.91243e-005 9.3243e-006 2.75508e-005 2.16734e-006 6.05094e-006 1.382e-005 8.73683e-006 ...
2.74438e-005 1.26201e-005 5.88439e-006 4.37434e-005 1.50278e-005 1.44923e-005 7.14471e-006 5.86737e-006 1.88955e-005 ...
7.76709e-006 1.5699e-005 1.53877e-005 2.01341e-005 1.58972e-005 1.37905e-006 1.33159e-005 2.57065e-006 6.28585e-006 ...
3.48481e-006 3.49193e-006 2.12925e-006 3.57633e-006 3.87287e-006 1.65707e-005 4.09129e-006 5.43498e-006 3.28955e-006 ...
4.7745e-006 5.94458e-006 8.49869e-006 1.09155e-005 2.93096e-006 9.11214e-007 1.456e-005 5.77817e-006 2.04244e-005 ...
6.3001e-006 1.00627e-005 1.8097e-005 9.13713e-007 1.37499e-005 7.00374e-007 8.07905e-007 1.26114e-005 1.89574e-006 ...

3.30889e-005 9.30541e-006 4.06423e-005 3.6721e-005 3.0455e-006 3.14951e-005 1.23849e-005 6.35112e-005 1.20773e-005 ...
1.18359e-005 1.4834e-006 1.9484e-005 3.30889e-005 9.30541e-006 4.06423e-005 3.6721e-005 3.0455e-006 3.14951e-005 ];
err_qmoy70=[...
1.6585e-006 1.7326e-005 5.49712e-006 2.43912e-006 3.3274e-006 5.037e-006 3.51277e-006 1.40926e-005 3.65947e-006 ...
2.18261e-006 3.53183e-006 8.30766e-006 6.21404e-006 1.96753e-005 7.14203e-006 4.74364e-006 5.33572e-006 2.10533e-005 ...
2.26257e-006 2.04382e-006 1.69894e-006 9.42819e-007 9.21499e-007 4.79159e-006 1.42568e-005 3.78009e-006 1.51096e-006 ...
5.80505e-006 6.02085e-006 4.31836e-006 1.11042e-005 5.77768e-006 2.86496e-006 5.16551e-006 7.33953e-006 1.0583e-005 ...
3.7852e-006 4.36205e-006 6.91595e-006 7.07543e-006 9.7775e-006 5.91516e-006 4.06799e-006 5.11137e-006 9.18648e-006 ...
3.02273e-006 3.9566e-006 1.18642e-005 5.41398e-006 4.3053e-006 2.76275e-006 4.35184e-006 1.78557e-006 8.10097e-006 ...
1.36943e-006 4.06789e-006 2.10466e-006 2.26409e-006 7.7104e-007 1.31103e-005 2.23341e-006 1.3338e-006 1.30542e-006 ...
3.32786e-006 2.09208e-006 1.83802e-005 6.1285e-006 1.29871e-006 5.52981e-006 2.23452e-005 4.7862e-006 3.41057e-006 ...
2.22579e-006 5.4521e-006 1.21934e-005 8.96233e-007 1.16899e-005 5.83715e-007 8.05887e-007 1.22796e-005 1.84094e-006 ...
1.69833e-005 3.68824e-006 8.80617e-006 1.61474e-005 3.9167e-006 8.58452e-006 6.2915e-006 5.59207e-005 9.41288e-006 ...
5.02393e-006 2.62667e-006 1.44055e-005 1.69833e-005 3.68824e-006 8.80617e-006 1.61474e-005 3.9167e-006 8.58452e-006 ];
err_qmoy60=[...
7.56964e-007 1.22835e-005 1.15229e-006 9.41093e-007 3.80965e-006 3.52693e-006 2.63334e-006 2.42558e-006 2.65146e-006 ...
1.43923e-006 3.61161e-006 3.46586e-006 2.94296e-006 5.92231e-006 3.21827e-006 3.76772e-006 2.90316e-006 1.45941e-005 ...
4.22834e-006 2.33568e-006 1.30689e-006 1.0969e-006 4.65804e-011 7.48145e-006 9.95272e-006 1.0218e-006 1.04148e-006 ...
5.76799e-006 2.58533e-006 3.02541e-006 3.24598e-006 2.18873e-006 8.90528e-007 3.91965e-006 4.91097e-006 7.19862e-006 ...
1.32433e-006 1.49603e-006 6.35198e-006 6.14978e-006 1.02122e-008 4.25125e-006 5.23499e-006 4.74297e-006 4.60745e-006 ...
3.20655e-006 2.69772e-006 9.56454e-009 3.53855e-006 1.71387e-006 1.01162e-006 7.10625e-006 1.37688e-006 8.41323e-006 ...
1.29345e-006 3.83556e-006 2.55449e-006 2.21221e-006 5.96816e-007 3.16872e-009 4.26785e-006 1.6232e-006 1.78424e-006 ...
2.3843e-006 1.75934e-006 9.85828e-006 2.73221e-006 2.38541e-006 1.93354e-006 9.09565e-006 4.31778e-006 3.31145e-006 ...
1.62932e-006 4.66648e-006 4.76157e-006 3.09294e-007 4.4938e-006 3.59103e-006 2.98189e-006 1.41384e-005 6.58827e-007 ...
6.83243e-006 7.89225e-009 4.5345e-006 1.46802e-005 2.24961e-006 4.64699e-006 5.60413e-006 7.90468e-005 5.14726e-006 ...
4.49991e-006 2.95458e-006 1.27401e-005 6.83243e-006 7.89225e-009 4.5345e-006 1.46802e-005 2.24961e-006 4.64699e-006 ];
err_qmoy50=[...
1.46378e-006 7.59816e-006 8.86019e-007 3.40087e-007 3.84051e-006 1.41851e-006 2.64041e-006 7.84651e-006 2.83452e-006 ...
8.98034e-007 8.43585e-007 5.38402e-006 2.54317e-006 6.53188e-009 3.05531e-006 3.31715e-006 3.64866e-006 6.87488e-006 ...
2.78096e-006 4.8437e-007 1.18169e-006 1.04505e-006 4.02507e-006 4.47091e-006 4.56619e-006 2.70612e-006 1.06226e-006 ...
2.82014e-006 4.47804e-006 2.01372e-006 1.08403e-005 3.76107e-006 2.30486e-006 3.97119e-006 2.24954e-006 2.46327e-008 ...
1.1309e-006 1.15181e-006 3.43316e-006 1.71259e-006 5.00964e-011 3.62457e-006 5.12507e-006 1.9053e-006 4.23083e-006 ...
1.33585e-006 2.0302e-006 1.09346e-008 4.17046e-006 7.22543e-007 1.23139e-006 1.72643e-006 5.44857e-007 3.85882e-006 ...
1.45425e-006 3.74188e-006 1.77799e-006 1.41817e-006 4.51225e-007 5.23947e-009 1.41841e-006 1.34275e-007 1.48388e-006 ...
2.4099e-006 1.51594e-006 7.66592e-006 2.39416e-006 1.7058e-006 4.0438e-006 9.86642e-006 1.47753e-006 2.01785e-006 ...
1.42918e-006 4.60403e-006 4.69847e-006 2.37528e-007 4.27322e-011 3.07279e-007 3.21971e-007 1.44997e-005 3.55064e-006 ...
4.42773e-006 1.6278e-010 5.11632e-006 5.57776e-006 9.00137e-007 3.6166e-006 6.04098e-006 5.4652e-005 6.89668e-006 ...
4.37688e-006 1.72978e-006 9.25383e-006 4.42773e-006 1.6278e-010 5.11632e-006 5.57776e-006 9.00137e-007 3.6166e-006 ];
err_qmoy40=[...
8.19452e-008 6.78312e-009 5.67717e-007 1.59861e-007 1.58809e-006 9.7457e-007 7.37253e-008 7.51627e-006 1.6539e-006 ...
7.49711e-008 5.71851e-010 7.18849e-006 2.5215e-006 7.25286e-006 2.08215e-006 7.36714e-007 6.41012e-007 4.13659e-006 ...
4.44444e-008 4.84369e-007 9.45181e-007 1.04503e-006 2.56823e-006 1.53939e-006 4.35849e-006 4.01725e-007 6.66211e-008 ...
2.16242e-009 6.1867e-007 1.47263e-006 7.93867e-006 1.61543e-006 3.34772e-006 7.7028e-007 2.61603e-006 2.33213e-008 ...
1.28637e-006 1.16504e-006 9.57337e-007 1.64777e-006 4.66122e-011 3.52283e-006 4.21196e-007 2.91909e-006 2.49781e-006 ...
6.42388e-007 2.22511e-006 1.09325e-008 3.57507e-006 8.46353e-007 1.23655e-008 1.30238e-006 9.32578e-007 5.10603e-011 ...
1.44544e-006 1.55081e-007 6.12561e-007 1.80056e-006 5.87933e-008 4.60825e-009 5.50691e-007 6.26275e-008 1.41389e-008 ...
1.62114e-006 1.33148e-006 7.66592e-006 2.37437e-006 1.23694e-006 7.14287e-007 3.64575e-006 1.60512e-008 1.29525e-006 ...
1.29511e-006 1.71928e-008 4.0362e-006 2.29259e-006 2.21448e-009 2.57869e-007 2.26875e-007 1.35573e-005 1.6385e-006 ...
2.75317e-006 4.40511e-011 4.83916e-006 3.72271e-006 2.75363e-007 3.16377e-006 5.74452e-006 5.03324e-005 2.36166e-006 ...
2.45339e-006 1.61006e-006 9.2169e-006 2.75317e-006 4.40511e-011 4.83916e-006 3.72271e-006 2.75363e-007 3.16377e-006 ];
err_qmoy30=[...
7.45827e-008 3.94816e-011 3.8458e-007 7.88079e-008 2.42644e-008 4.46967e-007 5.92646e-008 4.10151e-011 1.06297e-006 ...
4.62996e-008 4.29138e-011 2.31443e-006 3.87591e-006 4.14025e-011 8.76938e-007 8.99694e-007 2.20921e-011 6.29497e-006 ...
4.6143e-008 4.84344e-007 8.79362e-007 1.06113e-006 2.56478e-009 7.12148e-007 4.35844e-006 2.69552e-007 4.58454e-008 ...
1.26225e-008 4.97103e-007 6.71978e-007 2.46987e-009 1.65127e-006 2.25412e-006 2.03333e-008 1.2681e-006 2.18102e-008 ...
4.70931e-007 5.31297e-007 4.50777e-009 8.67201e-007 4.21552e-011 1.10775e-006 4.57518e-007 3.01428e-008 1.05519e-006 ...
5.99212e-007 1.07703e-006 3.97062e-011 5.04861e-007 8.21421e-008 1.23655e-008 2.04723e-006 1.07497e-007 5.06724e-011 ...
1.85892e-007 3.15348e-007 5.09711e-011 4.81637e-007 5.96495e-008 3.90398e-011 2.82835e-007 6.17492e-008 8.51241e-009 ...
1.67098e-006 3.71285e-008 1.08793e-006 1.30673e-006 1.23366e-006 3.69072e-008 3.09044e-006 5.3757e-009 1.13554e-006 ...
3.02189e-008 1.80032e-008 2.58465e-006 1.95392e-007 3.83451e-011 1.23911e-007 1.17596e-007 5.70234e-011 1.02059e-006 ...
1.08415e-006 4.16147e-011 1.25407e-006 2.49332e-006 1.83006e-011 1.33711e-006 3.43075e-006 5.16393e-005 2.31784e-006 ...
6.59839e-007 1.2621e-006 9.11159e-006 1.08415e-006 4.16147e-011 1.25407e-006 2.49332e-006 1.83006e-011 1.33711e-006 ];
err_qmoy20=[...
1.46817e-008 4.09832e-011 1.70771e-007 5.22749e-007 2.76785e-008 7.87563e-008 5.32581e-008 4.1965e-011 4.87751e-008 ...
3.89853e-008 6.72106e-010 1.31657e-007 3.03764e-008 4.21376e-011 3.45775e-007 3.43649e-008 2.36557e-011 1.43788e-006 ...
4.63835e-008 4.84344e-007 8.43138e-007 1.00299e-006 4.64295e-009 2.24132e-007 4.35844e-006 8.49603e-008 5.12007e-008 ...

```
1.36161e-008 1.85286e-007 1.68779e-007 4.70952e-011 2.21585e-007 1.24583e-011 1.65275e-009 2.37555e-007 2.18102e-008 ...
3.06464e-007 1.49935e-007 1.79559e-009 3.69543e-007 4.18526e-011 2.06098e-007 1.99368e-007 2.07975e-008 5.01349e-007 ...
3.1856e-007 7.52025e-008 3.94488e-011 1.45427e-008 8.00121e-008 1.23655e-008 5.91572e-008 8.56815e-009 4.93253e-011 ...
2.76466e-020 9.28905e-009 5.08819e-011 5.72817e-009 5.90364e-008 3.97228e-011 9.09949e-008 5.88363e-008 1.85737e-009 ...
1.47889e-007 4.23794e-008 1.08793e-006 3.96056e-007 1.05196e-006 3.1654e-008 1.02963e-006 4.44911e-011 6.54284e-007 ...
1.54668e-008 1.09294e-008 4.48878e-007 1.98686e-007 3.88989e-011 8.50257e-008 1.10897e-007 1.05403e-006 1.00881e-006 ...
7.81111e-007 4.25878e-011 2.21668e-007 8.44892e-007 1.8566e-011 4.73184e-007 9.51214e-007 4.95914e-005 5.75742e-008 ...
6.31565e-007 1.2621e-006 8.41083e-008 7.81111e-007 4.25878e-011 2.21668e-007 8.44892e-007 1.8566e-011 4.73184e-007 ];
err_qmoy10=[...
1.65706e-009 3.85642e-011 0 4.82021e-009 9.84922e-009 8.36671e-009 2.82426e-008 4.02424e-011 5.93651e-009 ...
2.23143e-008 2.65101e-011 2.18559e-008 2.96879e-008 4.01366e-011 9.0298e-010 3.51177e-008 1.93997e-011 7.82083e-007 ...
2.81786e-008 4.84344e-007 9.58533e-008 1.35483e-009 2.75583e-009 9.12021e-008 4.35844e-006 8.01232e-008 3.02337e-008 ...
8.93315e-011 6.86868e-008 8.07123e-007 4.80965e-011 2.33426e-008 8.89303e-026 9.47184e-010 1.6459e-007 2.18102e-008 ...
2.71943e-008 8.82368e-008 2.18873e-009 2.46532e-007 4.07037e-011 7.81841e-009 1.85973e-008 3.06299e-011 2.33362e-007 ...
2.14471e-008 4.77885e-008 3.95218e-011 1.30959e-008 4.69659e-008 2.23698e-011 3.2762e-008 4.30525e-009 5.01675e-011 ...
2.21726e-020 5.73913e-009 4.78228e-011 5.70714e-009 2.29421e-008 3.95492e-011 6.62417e-009 2.92949e-008 2.83796e-011 ...
1.89006e-008 2.93519e-008 1.08793e-006 6.92695e-007 0 2.53564e-008 1.19812e-007 4.35035e-011 9.03345e-008 ...
1.47305e-008 8.39811e-011 3.6508e-007 1.70558e-007 3.82059e-011 8.43004e-008 7.91247e-008 4.98156e-011 2.38981e-007 ...
4.42229e-008 4.16353e-011 1.00504e-008 4.58778e-008 1.33241e-011 1.77559e-008 2.44636e-008 4.66206e-005 5.49144e-008 ...
1.60544e-008 1.10658e-006 5.44626e-008 4.42229e-008 4.16353e-011 1.00504e-008 4.58778e-008 1.33241e-011 1.77559e-008 ];
```

```
% calcul de l'intervalle de confiance selon la distribution de Student
err_QMoy_G=[Stu_Const*std(err_qmoy95)/sqrt(length(err_qmoy95))...
        Stu_Const*std(err_qmoy90)/sqrt(length(err_qmoy90))...
        Stu_Const*std(err_qmoy80)/sqrt(length(err_qmoy80))...
        Stu_Const*std(err_qmoy70)/sqrt(length(err_qmoy70))...
        Stu_Const*std(err_qmoy60)/sqrt(length(err_qmoy60))...
        Stu_Const*std(err_qmoy50)/sqrt(length(err_qmoy50))...
        Stu_Const*std(err_qmoy40)/sqrt(length(err_qmoy40))...
        Stu_Const*std(err_qmoy30)/sqrt(length(err_qmoy30))...
        Stu_Const*std(err_qmoy20)/sqrt(length(err_qmoy20))...
        Stu_Const*std(err_qmoy10)/sqrt(length(err_qmoy10))];
% calcul de moyenne
moy_QMoy_G=[mean(err_qmoy95) mean(err_qmoy90) mean(err_qmoy80)...
mean(err_qmoy70) mean(err_qmoy60) mean(err_qmoy50) mean(err_qmoy40)...
mean(err_qmoy30) mean(err_qmoy20) mean(err_qmoy10) ];%calcul des moyennes
```

## B.2.4 Listing du programme « devC.m » contenant les données des comparaisons numérique.

```
err_max95=[...
0.00818634 0.00041003 0.0056759 0.00822039 0.000306455 0.00669507 0.0256346 0.000929023 0.0132716 ...
0.0148133 0.000391276 0.00988154 0.0256346 0.000929023 0.0132716 0.0148133 0.000391276 0.00988154 ...
0.00963341 0.00573233 0.00538868 0.0113973 7.06222e-006 0.0185232 0.0226554 0.000621907 0.00355322 ...
0.0152931 0.00108563 0.0056553 0.0103657 0.000946217 0.0050344 0.00934056 0.000660013 0.0116064 ...
0.00930113 0.000659271 0.00618977 0.00884766 0.000332377 0.020681 0.0142233 0.000647877 0.00580591 ...
0.00135473 0.000641866 0.0151697 0.000937242 0.00372458 0.022068 0.00100514 0.00683563 0.0118446 ...
0.00171485 0.00398903 7.47884e-005 0.00139511 0.0112752 0.00104809 0.00537438 0.0174571 0.00128815 ...
0.0199551 0.000853609 0.00593871 0.00870433 0.000572672 0.0129512 0.00929236 0.0261765 0.00141448 ...
0.00868406 0.00974105 0.00062838 0.0194403 0.00860212 0.000790113 0.00612506 0.00732243 0.0008266 ...
0.00473522 0.0085775 0.00113959 0.0045406 0.000403359 0.000382272 0.0149731 0.000918133 0.00728676 ...
0.0096535 0.000833705 0.0301859 0.0179542 0.00134491 0.0106293 0.00743071 0.000777921 0.0122487 ];
err_max90=[...
0.00290105 0.00109137 0.00364849 0.00822039 0.000982228 0.0045772 0.0132935 0.00133929 0.0145307 ...
0.0104366 0.000238904 0.00824328 0.0132935 0.00133929 0.0145307 0.0104366 0.000238904 0.00824328 ...
0.00650612 0.00428004 0.0032409 0.00426691 7.06222e-006 0.00590032 0.00443246 0.00113494 0.00219149 ...
0.0117885 0.000448966 0.00396025 0.00771609 0.00126187 0.00271358 0.00651545 0.000159878 0.00457632 ...
0.00686206 0.000701309 0.00258516 0.00670124 0.000332377 0.00531196 0.00681286 0.000562618 0.00249537 ...
0.000278901 0.000624037 0.0151697 0.00111613 0.00237923 0.00520852 0.000952506 0.00604397 0.0069544 ...
0.00136736 0.00349815 8.40423e-007 0.000778832 0.00602918 0.000982157 0.00363872 0.0098623 0.00087054 ...
0.0199527 0.000761671 0.00282085 0.00375273 0.000960936 0.00687016 0.00579639 0.0145026 0.00116737 ...
0.00707084 0.00739604 0.00046916 0.0173989 0.00383451 0.0010748 0.00267815 0.00297679 0.000825971 ...
0.0023327 0.00820104 0.000440027 0.0031871 0.000198339 0.000555995 0.0149731 0.000794772 0.00448097 ...
0.00595771 0.00105461 0.00859977 0.00672413 0.000659241 0.0030974 0.00743071 0.000594651 0.00425348 ];
err_max80=[...
0.00169499 0.00109137 0.00187766 0.00382809 0.000426324 0.000814126 0.0133329 0.000664376 0.00947453 ...
0.00608493 0.000110952 0.00463687 0.0133329 0.000664376 0.00947453 0.00608493 0.000110952 0.00463687 ...
0.00406775 0.00406092 0.00293709 0.00190068 7.06222e-006 0.0028337 0.00317707 0.00113494 0.000940568 ...
0.00317707 0.00104834 0.000709911 0.00310378 0.000881284 0.00213129 0.00480661 0.000389786 0.00134876 ...
0.00176978 0.000500931 0.00207058 0.00295521 0.000224271 0.00342538 0.00230299 0.000702738 0.00214494 ...
2.57605e-005 0.0016033 0.00276558 0.000831968 0.00147906 0.00130263 0.00110569 0.00253804 0.00643554 ...
0.00111695 0.00205722 2.2391e-005 0.000307338 0.00396706 0.000623633 0.00237827 0.00642644 0.0018032 ...
0.00460933 0.000669825 0.0013839 0.00139362 0.000171092 0.00446534 0.00144391 0.00339427 0.000727664 ...
0.00712514 0.00360381 0.000206246 0.00479169 0.00146371 0.000606584 0.00136073 0.00162865 0.00116654 ...
0.00172465 0.00409223 0.000991985 0.00293047 0.000178096 0.000751323 0.0149731 0.00182252 0.00352056 ...
0.00461792 0.000172354 0.00859977 0.00382825 0.00090589 0.00196118 0.00383645 0.000684905 0.00133825 ];
err_max70=[...
0.00117358 0.00109137 0.00187766 0.00165553 0.000279559 0.00059875 0.00433351 0.000668476 0.00391153 ...
0.00368388 0.000310762 0.00463687 0.00433351 0.000668476 0.00391153 0.00368388 0.000310762 0.00463687 ...
0.00179836 0.00487303 0.0013063 0.00107615 7.06222e-006 0.0028337 0.00317707 0.000776617 0.00032954 ...
0.00317707 0.000535263 0.000554303 0.000667734 0.00106311 0.00127815 0.000569349 0.000372515 0.00106798 ...
0.000946686 0.00116551 0.00207058 0.00110185 0.000377023 0.000935152 0.000581712 0.000769783 0.00214494 ...
0.00018636 0.000990775 0.00276558 0.000778598 0.00147906 0.000497131 0.00125453 0.00253804 0.00314873 ...
0.000844548 0.00135125 4.09176e-009 0.000582715 0.0030379 0.000842671 0.00151536 0.00358745 0.0018032 ...
0.00473843 0.000673416 0.000770427 0.000696903 0.000314451 0.00141539 0.000852189 0.00141995 0.000772973 ...
0.00707084 0.00183059 9.96074e-005 0.00539126 0.00146371 0.00140908 0.000857879 0.000751636 0.00138276 ...
0.00150284 0.000876825 0.000991985 0.00293047 1.69993e-005 0.000550656 0.00497191 0.000853157 0.00352056 ...
0.00108796 0.000172354 0.00812329 0.00129863 0.000961638 0.00118876 0.000995968 9.92734e-005 0.00133825 ];
err_max60=[...
0.00116825 0.00109137 0.000427326 0.00119669 0.000279559 0.000383574 0.00477576 0.00101028 0.00279114 ...
0.00118652 0.00013789 0.00172942 0.00477576 0.00101028 0.00279114 0.00118652 0.00013789 0.00172942 ...
0.00100282 0.00487303 0.000752175 0.000674662 7.06222e-006 0.00147817 0.00317707 0.000792835 0.00032954 ...
0.00179153 0.000417969 0.000416282 0.000164145 0.000687711 0.000208491 0.000354769 0.0002265 0.000553343 ...
0.000236807 0.00116935 0.000844956 0.000228238 0.000110643 0.000910056 9.73096e-005 0.000892637 0.00214494 ...
7.1391e-005 0.000288326 0.00277381 0.000778598 0.00147906 0.000497131 0.000949185 0.00184764 0.00314873 ...
0.000759393 0.00135125 1.75807e-009 0.000725079 0.0030379 0.000842671 0.00135125 0.00303481 0.000976499 ...
0.0030379 0.00103388 0.000558422 0.000386719 0.000398415 0.00141539 0.000654404 0.000471151 0.00132745 ...
0.00512499 0.000848942 8.61058e-005 0.00174054 0.00115754 0.000870474 0.000718705 0.000115343 0.000421813 ...
0.000626681 0.000795733 0.000335084 0.00293047 3.55886e-005 0.000751323 0.00437854 0.000853157 0.00352056 ...
0.000733947 0.000773268 0.00811851 0.000640223 0.000961638 0.000927773 0.000440018 0.000547366 0.00121002 ];
```

```
err_max50=[...
0.00104945 0.00109137 0.000427326 0.00040083 0.000155379 0.00032098 0.00284979 0.00101028 0.00269589 ...
0.000419655 5.94498e-005 0.00142451 0.00284979 0.00101028 0.00269589 0.000419655 5.94498e-005 0.00142451 ...
0.00033612 0.00425599 0.000654152 0.000429512 6.91267e-006 0.00147817 0.00244061 0.000622287 8.19378e-005 ...
0.00179153 0.000417969 0.000239866 0.000116933 0.000566683 0.00108879 0.000297722 0.000372224 0.000316903 ...
0.00012958 0.000810937 0.000659289 0.000216327 0.0001616 0.000910056 0.000110094 0.000306259 0.00214494 ...
0.000110731 0.00020528 0.00276558 0.000620753 0.00145261 0.0001035 0.000949185 0.00109861 0.00314873 ...
0.000759393 0.00135125 1.00414e-009 0.000725079 0.00213201 0.000842671 0.00135125 0.00303481 0.000976499 ...
0.00234587 0.000673416 0.000558422 0.000545608 0.000398415 0.00141539 0.00015595 0.000465743 0.000772973 ...
0.00298594 0.000478802 8.61058e-005 0.00172415 3.59007e-005 0.000870474 0.000309317 0.000115343 0.000421813 ...
0.000369994 0.000104871 0.000417986 0.00229886 7.00355e-007 0.000407757 0.00437854 0.00017207 0.00196977 ...
0.000122756 0.000344731 0.00333317 0.000169963 0.000679379 0.000636094 0.000189707 1.29984e-006 0.00121002 ];
err_max40=[...
8.1297e-005 0.00109137 9.66644e-005 8.1297e-005 0.000143277 0.00032098 0.00199075 0.00101028 0.00269589 ...
0.000342863 3.36617e-005 0.00130461 0.00199075 0.00101028 0.00269589 0.000342863 3.36617e-005 0.00130461 ...
3.49568e-005 0.00487303 0.000576678 0.000151611 6.93227e-006 0.00102673 0.00168447 0.00107416 8.19378e-005 ...
0.00167316 0.000407831 0.000171017 0.000598191 0.000687711 0.000209339 0.000147144 0.000372224 0.000655028 ...
2.746e-005 0.000810937 0.000341831 0.000216327 0.0001616 0.00124299 8.23181e-011 0.000221308 0.00214494 ...
7.46916e-005 0.000873307 0.00133616 0.000580813 0.00147711 3.00061e-006 9.31323e-010 0.00078471 0.00173861 ...
0.000189681 0.00135125 2.03965e-010 0.000725079 0.00213201 0.000842671 0.00135125 0.000240503 0.000976499 ...
0.00234587 0.000552924 0.000558422 0.000545608 1.72206e-006 0.000270756 0.000654404 0.000465743 0.000432688 ...
0.00298594 0.000478802 1.32551e-007 0.00129933 4.69615e-005 0.000870474 0.000229238 0.000115343 0.000421813 ...
0.000210885 0.000261757 0.000521893 0.00223311 5.18088e-005 0.000574767 0.00264819 0.000337669 0.00196977 ...
0.000201165 0.00127835 0.00333317 0.000171058 0.000679379 0.000636094 0.000132914 5.84205e-008 0.00129872 ];
err_max30=[...
0.00104945 0.00109137 2.27384e-006 0.000315456 0.000143277 0.000310475 0.00199075 0.000376467 0.00269589 ...
0.000350006 3.36617e-005 0.00123939 0.00199075 0.000376467 0.00269589 0.000350006 3.36617e-005 0.00123939 ...
3.85046e-005 0.00487303 0.000576678 0.000311516 6.93227e-006 0.00102673 0.000404588 0.00107416 8.19378e-005 ...
0.000214714 0.000307929 7.23757e-005 8.23181e-011 0.000446157 2.25995e-005 7.91805e-005 0.000365321 0.000655028 ...
1.06692e-005 0.000193629 0.00030933 3.5408e-006 2.68218e-005 0.000910056 4.1159e-011 0.000209397 0.00143562 ...
8.23181e-011 0.000873307 0.00133616 0.000620753 0.000884678 8.23181e-011 0.000205239 0.00078471 0.00161559 ...
9.77333e-007 0.00112128 5.29396e-023 0.000725079 0.00234587 0.000842671 0.00119295 0.000695369 0.000976499 ...
0.00234587 0.000552924 0.000318028 6.53597e-007 1.01156e-006 0.000270756 3.29971e-006 3.94904e-005 0.000279808 ...
0.000730036 0.000123332 4.65661e-010 0.00129933 4.69615e-005 0.000870474 5.52614e-005 0.000115343 0.000421813 ...
0.000181823 0.000156436 0.000521893 0.00128708 0.000180085 1.14063e-009 0.00183807 0.000156101 0.00104073 ...
3.93636e-005 0.00046169 0.00206481 0.00018799 0.000734261 0.00050803 5.04014e-005 4.65661e-010 0.00129872 ];
err_max20=[...
0.000150167 0.000238183 1.3475e-005 0.000315456 0.000143277 5.67134e-006 1.14344e-005 0.000135159 0.00236598 ...
6.00304e-006 4.65661e-010 0.00130461 1.14344e-005 0.000135159 0.00236598 6.00304e-006 4.65661e-010 0.00130461 ...
3.85046e-005 0.00487303 0.000766299 0.000263729 6.93227e-006 0.000913244 1.84112e-007 0.00018629 1.5221e-006 ...
1.58872e-007 0.00015927 0.00011839 1.49757e-006 0.000446157 2.49255e-006 5.55112e-017 4.65661e-010 0.000655028 ...
5.1565e-005 0.000485726 0.00030933 9.4242e-005 6.58545e-010 0.000688723 0 0.000192378 0.00143562 ...
0 0.000535183 0.00133611 0.000620753 0.000867476 0 9.31323e-010 0.00078471 0.00107915 ...
7.36275e-010 0.000595455 5.29396e-023 1.16914e-007 0.00210076 0.000110406 0.00062914 5.00669e-006 0.000137238 ...
0.00210263 0.000552924 0.000418878 4.1159e-011 6.17828e-007 0.000270756 5.82077e-011 5.82077e-011 0.000279808 ...
1.55811e-005 2.4013e-006 3.29272e-010 0.000406105 3.59007e-005 0.000870474 5.52614e-005 0.000115343 0.000421813 ...
0.000181823 1.16415e-010 0.000149389 0.000961342 0 1.14063e-009 0.00183807 9.31323e-010 0.000781406 ...
0 1.14063e-009 0.00112228 0 1.92606e-005 0.000173992 0 4.65661e-010 0.000321772 ];
err_max10=[...
1.15196e-007 8.46526e-006 6.30533e-008 7.80196e-005 0.000143277 1.87405e-005 2.40565e-008 2.34017e-005 0.00140869 ...
8.23181e-011 3.29272e-010 0.000789905 2.40565e-008 2.34017e-005 0.00140869 8.23181e-011 3.29272e-010 0.000789905 ...
3.75509e-006 0.00487303 4.51366e-005 0.000232393 6.90081e-006 0.000270199 1.59764e-007 9.31323e-010 5.07323e-007 ...
6.13306e-008 6.58545e-010 1.18951e-006 1.49757e-006 8.98595e-005 2.19558e-006 5.55112e-017 3.29272e-010 0.000305408 ...
0 3.84043e-005 0.000115617 0 6.58545e-010 0.000327868 0 2.39909e-005 0.000123784 ...
0 1.31709e-009 0.000532085 0.000570603 3.19404e-006 0 1.31709e-009 0.000439466 1.16415e-010 ...
6.58545e-010 0.000201132 5.29396e-023 9.31323e-010 0.000943235 1.01328e-006 0.000355885 8.15286e-005 0.000326759 ...
0.00105635 0.00010066 0.000249106 0 3.29272e-010 0.000132568 0 0 5.34298e-005 ...
3.04921e-006 4.1159e-011 2.32831e-010 2.76673e-005 3.59007e-005 0.00043594 5.52614e-005 0.000115343 5.63903e-005 ...
7.36786e-006 0 0.000149389 0.00051346 0 1.14063e-009 0.000513804 9.31323e-010 0.000243346 ...
0 1.31709e-009 0.000561181 0 8.06549e-010 6.87961e-005 0 3.29272e-010 0.00022001];
```

```
err_Max_C=[Stu_Const*std(err_max95)/sqrt(length(err_max95))...
Stu_Const*std(err_max90)/sqrt(length(err_max90))...
Stu_Const*std(err_max80)/sqrt(length(err_max80))...
Stu_Const*std(err_max70)/sqrt(length(err_max70))...
Stu_Const*std(err_max60)/sqrt(length(err_max60))...
Stu_Const*std(err_max50)/sqrt(length(err_max50))...
Stu_Const*std(err_max40)/sqrt(length(err_max40))...
Stu_Const*std(err_max30)/sqrt(length(err_max30))...
Stu_Const*std(err_max20)/sqrt(length(err_max20))...
Stu_Const*std(err_max10)/sqrt(length(err_max10))];
moy_Max_C=[mean(err_max95) mean(err_max90) mean(err_max80) mean(err_max70)...
mean(err_max60) mean(err_max50) mean(err_max40) mean(err_max30) mean(err_max20)...
mean(err_max10) ];%calcul des moyennes
```

```
err_moy95=[...
0.000336394 1.1393e-007 0.000217622 0.000348209 3.91157e-008 0.000242029 0.00114971 1.34584e-007 0.000266692 ...
0.000800647 7.00016e-008 0.000341236 0.00114971 1.34584e-007 0.000266692 0.000800647 7.00016e-008 0.000341236 ...
0.000565406 3.43241e-006 0.000225923 0.000587756 8.68795e-007 0.000338528 0.000348885 1.8081e-007 0.000104939 ...
0.000254089 1.83168e-007 0.000113748 0.000420143 1.97549e-007 0.000161156 0.000528786 1.66767e-007 0.000367485 ...
0.000761847 2.62118e-007 0.000131059 0.00093528 3.47628e-008 0.000919434 0.000637088 2.0973e-007 0.000173256 ...
3.01315e-006 1.52147e-007 0.000439608 2.88424e-007 0.000158764 0.000682848 2.46135e-007 0.0001624 0.00108116 ...
2.33373e-007 0.000265738 2.43907e-008 2.248e-007 0.000573391 3.48548e-007 0.000272626 0.000822322 1.97451e-007 ...
0.000625077 3.21296e-007 0.00019598 0.000431191 1.09467e-007 0.000241967 0.00053881 0.000927256 2.41512e-007 ...
0.000302613 0.000746787 1.68453e-007 0.000492345 0.00023421 2.18642e-007 0.000291761 0.000265718 1.25345e-007 ...
9.91418e-005 0.000645794 5.64158e-007 0.000248239 5.69453e-006 7.02513e-008 0.0003615 1.0487e-007 0.000291909 ...
0.000511902 1.12299e-007 0.000569628 0.000573236 3.25971e-007 0.000163459 0.000373786 2.67546e-007 0.000138243];
err_moy90=[...
9.40179e-005 4.30041e-007 5.1485e-005 0.000193555 1.78691e-007 0.000114332 0.000556608 3.06062e-007 9.55914e-005 ...
0.000495646 8.56806e-008 0.000170627 0.000556608 3.06062e-007 9.55914e-005 0.000495646 8.56806e-008 0.000170627 ...
0.000271977 2.54638e-006 8.17959e-005 0.000255873 1.32197e-006 0.000135044 3.92661e-005 2.83687e-007 2.38606e-005 ...
0.000100515 1.06078e-007 6.05639e-005 0.000219983 3.48779e-007 5.94631e-005 0.000192744 2.03504e-008 0.000124769 ...
0.000318425 3.0798e-007 4.59332e-005 0.000252909 4.45243e-008 0.000281148 0.000226805 1.89083e-007 5.51774e-005 ...
1.04749e-006 5.21697e-008 0.000299991 5.80077e-007 5.77757e-005 0.000138843 9.82013e-008 9.79349e-005 0.000400856 ...
4.28504e-007 9.9495e-005 4.83209e-010 8.26591e-008 0.000244771 4.28396e-007 0.000105044 0.000409659 1.42067e-007 ...
0.000411176 1.95881e-007 5.1404e-005 0.000120463 1.28724e-007 0.000106432 0.000188082 0.000447002 5.05523e-007 ...
9.57719e-005 0.000362169 1.18678e-007 0.00021814 7.82064e-005 3.60618e-007 7.45861e-005 6.77263e-005 3.99498e-007 ...
5.65472e-005 0.000281364 2.02929e-007 7.6358e-005 1.45059e-006 9.29803e-008 0.000202235 3.95254e-007 0.000110017 ...
0.000343943 2.48899e-007 0.000256279 0.000209307 1.92812e-007 4.3508e-005 0.000173594 1.74449e-007 5.1884e-005 ];

err_moy80=[...
2.05718e-005 5.02449e-007 1.64735e-005 1.93605e-005 1.10153e-007 3.86002e-005 0.000196017 3.09684e-007 3.16471e-005 ...
0.000207268 6.57207e-008 6.65496e-005 0.000196017 3.09684e-007 3.16471e-005 0.000207268 6.57207e-008 6.65496e-005 ...
0.00010393 1.70357e-006 2.22641e-005 4.95268e-005 9.05398e-007 4.33665e-005 1.01716e-005 4.34058e-007 2.89201e-006 ...
1.02181e-005 2.78802e-007 2.22471e-005 1.45567e-005 3.62096e-007 1.70046e-005 1.53498e-005 1.22685e-007 6.61431e-005 ...
3.02392e-005 1.71051e-007 9.17326e-006 4.24422e-005 7.87003e-008 9.44982e-005 1.46585e-005 1.90814e-007 2.28754e-005 ...
2.46217e-007 1.43705e-007 3.43656e-005 3.43734e-007 1.66301e-005 8.7347e-006 1.66064e-007 3.19708e-005 8.60901e-005 ...
3.43591e-007 3.81965e-005 2.81501e-009 5.33973e-008 0.000110922 2.70586e-007 3.66008e-005 8.50863e-005 3.25748e-007 ...
9.2723e-005 3.17179e-007 1.4688e-005 1.31019e-005 4.20231e-008 5.22325e-005 1.00267e-005 8.78971e-005 3.42178e-007 ...
3.12808e-005 6.5151e-005 8.59237e-008 0.000102709 1.13836e-005 3.06234e-007 8.42802e-006 9.76715e-006 4.88486e-007 ...
1.87732e-005 4.21121e-005 2.6116e-007 2.82028e-005 1.99814e-007 1.90734e-007 0.000144514 2.85176e-007 4.38455e-005 ...
0.000101678 4.02608e-008 0.000104718 3.21529e-005 2.45139e-007 9.31323e-006 1.8585e-005 1.29121e-007 1.00456e-005 ];

err_moy70=[...
9.3267e-006 3.54321e-007 1.16253e-005 5.94098e-006 6.85201e-008 7.41308e-006 4.95275e-005 3.50534e-007 1.45448e-005 ...
3.51745e-005 1.2762e-007 3.65484e-005 4.95275e-005 3.50534e-007 1.45448e-005 3.51745e-005 1.2762e-007 3.65484e-005 ...
1.69199e-005 2.4127e-006 7.21506e-006 1.0503e-005 1.69156e-006 1.77674e-005 6.66113e-006 5.69952e-007 1.10159e-006 ...
6.04959e-006 3.0169e-007 4.76755e-006 8.60592e-007 5.22928e-007 8.37701e-006 1.04295e-006 2.0414e-007 2.21153e-005 ...
4.83633e-006 4.35775e-007 4.80574e-006 4.59502e-006 1.51377e-007 3.30312e-005 8.952e-007 2.56234e-007 1.22943e-005 ...
8.07598e-008 2.23101e-007 2.03469e-005 2.93598e-007 1.08425e-005 5.71525e-007 1.26483e-007 1.92165e-005 1.66896e-005 ...
```

3.53144e-007 1.86571e-005 2.04697e-011 6.00713e-008 2.04584e-005 5.2696e-007 2.20117e-005 1.75893e-005 3.46925e-007 ...
4.26129e-005 3.26062e-007 6.70539e-006 1.99265e-006 4.04219e-008 1.62712e-005 2.04995e-006 1.95121e-005 4.45972e-007 ...
1.76386e-005 2.15458e-005 5.17354e-008 3.82147e-005 5.20234e-006 5.26019e-007 1.55722e-006 2.1736e-006 4.81577e-007 ...
5.32485e-006 4.33311e-006 2.82091e-007 1.3425e-005 6.03094e-008 9.59692e-008 4.64677e-005 2.16427e-007 2.38811e-005 ...
2.82437e-005 3.17475e-008 4.93297e-005 3.66957e-006 4.37468e-007 2.88139e-006 1.58059e-006 5.12359e-009 2.8486e-006];

err_moy60=[...
7.48961e-006 2.05746e-007 8.66749e-007 3.1714e-006 9.41111e-008 3.08178e-006 2.03057e-005 3.35285e-007 8.72902e-006 ...
8.50577e-006 7.09588e-008 1.27681e-005 2.03057e-005 3.35285e-007 8.72902e-006 8.50577e-006 7.09588e-008 1.27681e-005 ...
2.548e-006 1.6196e-006 2.67158e-006 2.30955e-006 2.16567e-006 8.7811e-006 6.1563e-006 5.07713e-007 9.38673e-007 ...
1.85758e-006 2.5727e-007 2.19193e-006 1.91319e-007 4.75713e-007 3.80778e-006 2.57076e-007 6.19073e-008 9.48011e-006 ...
6.3179e-007 4.02158e-007 2.72173e-006 1.04061e-006 1.89748e-008 1.37969e-005 8.17292e-008 2.8917e-007 9.82856e-006 ...
3.17238e-008 4.08577e-008 1.84346e-005 2.62561e-007 8.48528e-006 4.43318e-007 9.43677e-008 1.31498e-005 7.15312e-006 ...
3.12023e-007 1.13581e-005 3.81605e-012 6.07143e-008 1.41162e-005 3.98025e-007 1.55985e-005 6.43983e-006 3.06513e-007 ...
1.34649e-005 5.72034e-007 2.88399e-006 8.36948e-007 8.53839e-008 7.98044e-006 7.79716e-007 1.17758e-005 6.37469e-007 ...
7.27528e-006 8.25299e-006 4.65028e-008 1.59114e-005 6.99456e-007 3.69582e-007 6.17049e-007 1.84595e-007 1.71781e-007 ...
1.829e-006 1.09317e-006 2.16205e-007 9.13898e-006 2.53123e-008 1.57523e-007 2.03537e-005 2.09919e-007 1.77333e-005 ...
2.50003e-005 7.03664e-008 4.10829e-005 5.4885e-007 4.44215e-007 1.68575e-006 5.2484e-007 2.88079e-008 2.07884e-006 ];

err_moy50=[...
7.07969e-007 2.59603e-007 5.02663e-007 2.90076e-007 5.41032e-008 1.26659e-006 1.62178e-005 2.61891e-007 5.48086e-006 ...
4.24004e-006 5.33817e-008 6.86037e-006 1.62178e-005 2.61891e-007 5.48086e-006 4.24004e-006 5.33817e-008 6.86037e-006 ...
1.09802e-006 1.68989e-006 1.72281e-006 1.02777e-006 8.94494e-007 5.50478e-006 3.3674e-006 2.96625e-007 4.98157e-007 ...
1.67908e-006 1.28705e-007 1.04028e-006 1.10348e-007 4.17131e-007 1.52851e-006 1.18469e-007 7.98849e-008 4.10573e-006 ...
2.47895e-007 2.58334e-007 1.54776e-006 4.71476e-007 5.19254e-008 5.64339e-006 4.316e-008 1.84742e-007 9.16422e-006 ...
1.80508e-008 1.15236e-008 1.49156e-005 3.27042e-007 6.78269e-006 1.52008e-007 1.16758e-007 2.24741e-006 4.27368e-006 ...
2.58416e-007 4.86998e-006 4.72125e-013 5.76707e-008 8.10235e-006 2.89835e-007 3.9911e-006 2.84193e-006 2.27222e-007 ...
8.1914e-006 3.5324e-007 1.56594e-006 5.21148e-007 4.77389e-008 2.99963e-006 3.67936e-007 1.1028e-005 5.40504e-007 ...
2.30278e-006 5.65239e-006 2.12364e-008 6.07171e-006 4.23804e-008 3.27408e-007 1.49777e-007 1.18514e-007 9.6121e-008 ...
6.14032e-007 1.03746e-007 2.52632e-007 3.73582e-006 2.42847e-009 1.0306e-007 1.62394e-005 6.71016e-008 6.85695e-006 ...
2.27553e-007 3.74726e-008 1.78136e-005 9.47057e-008 3.08289e-007 9.48944e-007 1.5962e-007 1.10941e-010 1.81808e-006 ];

err_moy40=[...
9.88492e-008 3.02235e-007 1.66543e-007 9.16224e-008 2.0756e-008 4.37711e-007 2.23353e-006 1.44081e-007 3.85602e-006 ...
2.48996e-006 3.39373e-009 3.40832e-006 2.23353e-006 1.44081e-007 3.85602e-006 2.48996e-006 3.39373e-009 3.40832e-006 ...
4.59349e-007 1.17482e-006 1.0109e-006 4.94782e-007 7.69009e-007 1.92791e-006 1.84271e-006 5.66114e-007 2.82947e-007 ...
1.58132e-006 1.00094e-007 3.75992e-007 1.12688e-007 2.25663e-007 6.01229e-007 2.7107e-008 6.76444e-008 1.99399e-006 ...
1.34264e-008 2.31645e-007 7.03643e-007 4.29763e-008 3.3168e-008 3.78916e-006 1.47399e-013 5.17673e-008 8.09022e-006 ...
4.3004e-008 1.55752e-007 7.45003e-006 3.47498e-007 5.90249e-006 2.48645e-009 2.19184e-011 1.35993e-006 1.42627e-006 ...
1.20972e-008 2.43168e-006 6.73092e-014 5.76061e-008 6.50668e-006 2.02652e-007 3.05312e-006 4.93274e-007 2.46695e-007 ...
5.95131e-006 3.66739e-007 7.07421e-007 1.32519e-007 3.05615e-010 1.1359e-006 1.70229e-007 1.07138e-005 3.22029e-007 ...
1.28687e-006 5.07786e-006 3.62886e-011 2.35266e-006 4.4479e-008 2.85081e-007 1.30041e-007 1.17195e-007 1.40338e-007 ...
3.14515e-007 1.24649e-007 2.11229e-007 3.40238e-006 3.47543e-008 7.31237e-008 5.82649e-006 7.49422e-008 6.02123e-006 ...
9.97324e-008 1.16545e-007 1.56765e-005 1.56729e-007 3.8852e-007 7.91453e-007 4.21187e-008 1.11639e-011 1.35235e-006 ];

err_moy30=[...
4.41399e-007 3.22764e-007 6.93476e-008 1.5976e-007 2.07057e-008 2.60536e-007 1.1177e-006 7.5694e-008 1.77108e-006 ...
1.59964e-006 6.78498e-009 2.24538e-006 1.1177e-006 7.5694e-008 1.77108e-006 1.59964e-006 6.78498e-009 2.24538e-006 ...
1.72953e-007 1.46907e-006 4.74449e-007 3.12742e-007 4.7417e-007 1.51813e-006 1.26058e-007 5.8798e-007 9.53529e-008 ...
8.12478e-008 3.68581e-008 2.05217e-007 2.98012e-014 1.43177e-007 1.48412e-007 1.61902e-008 1.22328e-007 5.87983e-007 ...
5.0749e-009 3.04819e-008 2.93934e-007 1.32751e-009 4.15288e-009 2.18195e-006 1.09379e-014 5.62273e-008 4.42634e-006 ...
4.36473e-014 1.05059e-007 6.94994e-006 4.19683e-007 3.51216e-006 2.33464e-014 1.15257e-008 1.0289e-006 7.86708e-007 ...
1.02103e-010 1.5092e-006 1.35151e-027 5.76323e-008 5.13214e-006 1.48296e-007 1.47436e-006 3.81332e-007 1.04979e-007 ...
5.44805e-006 2.69322e-007 2.65631e-007 2.75753e-010 1.21344e-010 3.5525e-007 2.3424e-009 1.62733e-008 7.00652e-008 ...
4.98897e-007 2.63646e-008 5.7762e-012 1.07934e-006 4.28468e-008 2.59112e-007 5.32111e-008 1.15669e-007 8.26845e-008 ...
2.52467e-007 3.94631e-008 1.61378e-007 2.01711e-006 4.04064e-008 2.27432e-011 4.50367e-006 1.1377e-008 1.4085e-006 ...
1.92213e-008 5.51263e-008 3.43876e-006 4.55493e-008 1.91159e-007 2.07413e-007 2.45011e-008 6.75439e-012 6.66977e-007 ];

err_moy20=[...
9.6121e-008 2.86588e-008 8.19206e-009 3.01664e-007 1.38903e-008 3.58448e-008 5.19416e-009 4.43787e-008 1.12245e-006 ...
2.69245e-008 2.21429e-012 1.35496e-006 5.19416e-009 4.43787e-008 1.12245e-006 2.69245e-008 2.21429e-012 1.35496e-006 ...
2.65501e-008 1.54502e-006 3.8742e-007 6.8673e-008 4.7315e-007 1.19189e-006 4.63379e-009 3.33054e-008 1.57365e-008 ...
4.36395e-009 8.27735e-009 7.52839e-008 1.95146e-010 8.87435e-008 1.99559e-008 2.70617e-022 7.43716e-012 1.36731e-007 ...
1.5537e-008 7.06768e-008 1.5247e-007 1.52831e-008 2.5191e-012 6.35297e-007 0 2.92976e-008 4.3719e-006 ...
0 3.66272e-008 6.58667e-006 3.68394e-007 3.35461e-006 0 2.88956e-011 7.25751e-007 2.35623e-007 ...
1.08544e-011 6.02736e-007 2.98778e-028 5.68024e-011 4.12205e-006 7.89571e-009 6.81789e-007 2.39807e-008 1.11552e-008 ...

4.1212e-006 1.07276e-007 2.04749e-007 6.2418e-015 4.59738e-011 1.10979e-007 1.40172e-014 1.03342e-014 4.59962e-008 ...
3.70732e-008 2.55483e-010 5.5761e-012 2.08061e-007 2.54672e-008 2.04338e-007 5.54807e-008 1.15394e-007 5.28573e-008 ...
1.39769e-007 1.72036e-014 2.92409e-008 9.82357e-007 0 2.47342e-011 3.63576e-006 1.15759e-011 7.2177e-007 ...
0 2.53304e-011 2.54336e-006 0 1.22833e-009 1.00398e-007 0 6.36018e-012 2.55637e-007 ];

err_moy10=[...
2.88374e-010 1.2249e-009 1.81672e-011 1.35898e-008 1.38898e-008 3.3144e-009 4.07374e-012 1.5267e-009 5.18803e-007 ...
9.08222e-015 1.661e-012 5.23635e-007 4.07374e-012 1.5267e-009 5.18803e-007 9.08222e-015 1.661e-012 5.23635e-007 ...
1.22344e-009 1.38841e-006 1.85552e-008 4.51464e-008 3.74897e-007 1.80205e-007 1.61554e-010 9.1781e-012 1.61955e-009 ...
9.89021e-011 8.88731e-012 1.24662e-009 1.95135e-010 5.33681e-009 4.14359e-010 1.08189e-022 6.1181e-012 2.2437e-008 ...
0 4.87369e-009 6.05945e-008 0 2.05577e-012 1.27443e-007 0 1.78617e-008 1.63862e-008 ...
0 2.73398e-011 3.00248e-007 6.65934e-008 5.97141e-009 0 3.1222e-011 2.80503e-007 5.92667e-014 ...
1.20703e-011 1.12002e-007 2.98778e-028 3.09541e-011 9.5284e-007 7.2576e-010 2.76909e-007 1.15098e-008 2.33048e-008 ...
1.33855e-006 5.96749e-009 9.63273e-008 0 7.22118e-012 6.7261e-008 0 0 3.00579e-009 ...
1.44895e-009 6.85141e-015 4.62339e-012 8.25127e-009 2.51805e-008 5.96976e-008 5.54837e-008 1.15042e-007 8.4834e-009 ...
7.07468e-009 0 2.92415e-008 3.50303e-007 0 2.43112e-011 4.43997e-007 9.91527e-012 1.657e-007 ...
0 2.54e-011 6.54232e-007 0 8.32286e-012 2.66877e-008 0 5.44023e-012 4.08654e-008 ];

```
% calcul de l'intervalle de confiance selon la distribution de Student
err_Moy_C=[Stu_Const*std(err_moy95)/sqrt(length(err_moy95))...
        Stu_Const*std(err_moy90)/sqrt(length(err_moy90))...
        Stu_Const*std(err_moy80)/sqrt(length(err_moy80))...
        Stu_Const*std(err_moy70)/sqrt(length(err_moy70))...
        Stu_Const*std(err_moy60)/sqrt(length(err_moy60))...
        Stu_Const*std(err_moy50)/sqrt(length(err_moy50))...
        Stu_Const*std(err_moy40)/sqrt(length(err_moy40))...
        Stu_Const*std(err_moy30)/sqrt(length(err_moy30))...
        Stu_Const*std(err_moy20)/sqrt(length(err_moy20))...
        Stu_Const*std(err_moy10)/sqrt(length(err_moy10))];
moy_Moy_C=[mean(err_moy95) mean(err_moy90) mean(err_moy80) mean(err_moy70)...
mean(err_moy60) mean(err_moy50) mean(err_moy40) mean(err_moy30) mean(err_moy20)...
mean(err_moy10) ];%calcul des moyennes
```

err_qmoy95=[...
0.000904868 4.01346e-006 0.000600433 0.000876786 2.02185e-006 0.000432397 0.00227747 6.37251e-006 0.000619031 ...
0.00162575 3.13591e-006 0.000681095 0.00227747 6.37251e-006 0.000619031 0.00162575 3.13591e-006 0.000681095 ...
0.00126816 7.41967e-005 0.000580221 0.00142515 1.43962e-006 0.000824754 0.00170134 6.16462e-006 0.0003389 ...
0.000993232 8.13553e-006 0.000269907 0.00107455 7.0509e-006 0.000433752 0.00120662 6.42836e-006 0.000848931 ...
0.00146247 6.96673e-006 0.000408886 0.00179523 2.15764e-006 0.0018631 0.00161685 6.25662e-006 0.000514286 ...
1.66314e-005 5.62456e-006 0.0013466 9.18238e-006 0.000428265 0.00174602 8.37144e-006 0.000430984 0.00203091 ...
1.22895e-005 0.000588482 8.83453e-007 9.63375e-006 0.000987993 1.13882e-005 0.000656534 0.00178215 9.37979e-006 ...
0.00173489 9.20172e-006 0.000531344 0.00108983 4.11159e-006 0.000532905 0.00127419 0.00189465 9.61139e-006 ...
0.000688378 0.00151237 6.09459e-006 0.00106933 0.000813779 7.23166e-006 0.000726552 0.000871162 5.67003e-006 ...
0.000237925 0.00138243 1.3666e-005 0.000584404 1.73541e-005 3.19906e-006 0.00105751 4.97071e-006 0.000724215 ...
0.00101302 4.89984e-006 0.00194445 0.00175468 1.11056e-005 0.000524464 0.000923697 7.68042e-006 0.000568226 ];
err_qmoy90=[...
0.000289448 1.29256e-005 0.000200923 0.000631986 7.56711e-006 0.000230751 0.00118247 1.18043e-005 0.000302942 ...
0.00113427 2.80647e-006 0.000411847 0.00118247 1.18043e-005 0.000302942 0.00113427 2.80647e-006 0.000411847 ...
0.000654458 6.0919e-005 0.000258212 0.00063917 2.0727e-006 0.000325118 0.000215429 8.25732e-006 0.000114394 ...
0.000650949 3.88422e-006 0.00014563 0.000718101 1.10653e-005 0.000180242 0.00055292 1.11301e-006 0.000254702 ...
0.000691438 8.40049e-006 0.000169493 0.000560306 2.2132e-006 0.000570435 0.000707183 5.06671e-006 0.000192389 ...
4.79275e-006 3.10486e-006 0.00124239 1.46272e-005 0.000205312 0.000421669 5.8574e-006 0.000301976 0.000817904 ...
1.26302e-005 0.000271402 6.21683e-009 4.85133e-006 0.00053589 1.28248e-005 0.000295754 0.000904712 6.82104e-006 ...
0.00151063 6.36001e-006 0.000171332 0.000378173 5.38547e-006 0.000263729 0.000547945 0.000909585 1.43005e-005 ...
0.000288857 0.000835493 4.2268e-006 0.000558948 0.000308385 1.09699e-005 0.000279166 0.000236535 1.05833e-005 ...
0.00014197 0.000716668 5.01781e-006 0.00024607 7.0482e-006 4.68561e-006 0.00088262 9.50409e-006 0.00033799 ...
0.000695144 8.94005e-006 0.0006266 0.000602226 6.338e-006 0.000191148 0.00061489 5.68791e-006 0.000178261 ];

```
err_qmoy80=[...
0.000105024 1.3259e-005 8.71409e-005 0.00015078 3.59847e-006 8.59705e-005 0.000575144 8.21088e-006 0.000139697 ...
0.000517211 1.6676e-006 0.000226047 0.000575144 8.21088e-006 0.000139697 0.000517211 1.6676e-006 0.000226047 ...
0.000323069 4.75973e-005 0.000104368 0.000164121 1.4772e-006 0.000118833 0.000110648 1.1184e-005 1.94519e-005 ...
0.000112484 8.57455e-006 6.18288e-005 8.89963e-005 8.96965e-006 6.90873e-005 9.78121e-005 3.81989e-006 0.000133384 ...
0.000112248 5.31417e-006 4.96107e-005 0.000160553 2.74261e-006 0.000185541 9.7775e-005 5.10022e-006 0.000118249 ...
1.35598e-006 9.85201e-006 0.00016183 9.75023e-006 8.96597e-005 5.86413e-005 7.90382e-006 0.000130036 0.000289853 ...
8.45971e-006 0.000144635 1.74914e-007 2.38483e-006 0.000364813 7.73327e-006 0.000135971 0.000305338 1.28619e-005 ...
0.000285495 8.35444e-006 5.75122e-005 7.42897e-005 1.65841e-006 0.00011942 5.66882e-005 0.000232221 8.21887e-006 ...
0.000147996 0.00020492 2.29783e-006 0.000239105 7.42339e-005 6.63643e-006 6.32301e-005 6.7106e-005 1.37769e-005 ...
6.30234e-005 0.000167355 8.20445e-006 0.00013908 2.00351e-006 7.19441e-006 0.000847382 1.38089e-005 0.000205825 ...
0.000271413 1.55801e-006 0.000492181 0.000183161 7.29842e-006 7.23375e-005 0.000117425 5.05977e-006 4.23233e-005 ];
err_qmoy70=[...
6.91113e-005 1.22694e-005 7.19303e-005 5.48241e-005 2.27714e-006 2.63307e-005 0.000242635 9.22279e-006 7.86231e-005 ...
0.000139755 3.32002e-006 0.00020489 0.000242635 9.22279e-006 7.86231e-005 0.000139755 3.32002e-006 0.00020489 ...
8.52844e-005 5.60149e-005 4.51995e-005 5.60665e-005 2.6267e-006 7.03985e-005 0.000101287 1.15548e-005 7.0615e-006 ...
9.92409e-005 6.75686e-006 2.0153e-005 1.42608e-005 1.18662e-005 3.08725e-005 1.64474e-005 5.50943e-006 5.39889e-005 ...
3.45778e-005 1.03961e-005 3.01104e-005 3.23455e-005 4.77537e-006 7.24302e-005 1.51554e-005 7.42172e-006 8.593e-005 ...
2.06558e-006 9.37834e-006 0.000148891 8.90923e-006 7.69849e-005 8.08648e-006 7.40315e-006 0.000108013 0.000109663 ...
9.45652e-006 8.92113e-005 1.20111e-010 3.45913e-006 0.000105819 1.16493e-005 0.000102726 0.000119599 1.26877e-005 ...
0.000224367 7.14167e-006 2.39702e-005 1.36499e-005 2.15047e-006 4.38452e-005 1.62189e-005 7.17345e-005 9.26771e-006 ...
0.000115867 8.69754e-005 1.22823e-006 0.000114934 5.41203e-005 1.23289e-005 2.37776e-005 2.39368e-005 1.40837e-005 ...
3.38322e-005 3.52903e-005 7.60541e-006 9.98852e-005 3.63939e-007 3.98294e-006 0.000291784 7.09294e-006 0.000172249 ...
9.88866e-005 1.47374e-006 0.000381765 4.41365e-005 1.12111e-005 3.16949e-005 2.18813e-005 4.98759e-007 2.56734e-005 ];
err_qmoy60=[...
6.33813e-005 8.91757e-006 7.79164e-006 4.00797e-005 2.60614e-006 1.5495e-005 0.000150308 1.01678e-005 5.77337e-005 ...
4.26026e-005 1.82327e-006 4.42261e-005 0.000150308 1.01678e-005 5.77337e-005 4.26026e-005 1.82327e-006 4.42261e-005 ...
2.30827e-005 5.2972e-005 1.61674e-005 2.10263e-005 2.95473e-006 3.58145e-005 0.000100611 1.12753e-005 6.19426e-006 ...
3.77228e-005 5.94916e-006 1.22976e-005 3.33711e-006 9.44889e-006 1.09462e-005 5.81902e-006 2.5117e-006 2.737e-005 ...
6.53236e-006 1.02086e-005 1.56731e-005 8.94937e-006 9.75287e-007 3.79907e-005 1.44513e-006 8.52147e-006 8.41165e-005 ...
8.96621e-007 2.19107e-006 0.000147689 8.41297e-006 7.231e-005 7.07116e-006 5.77658e-006 9.01138e-005 8.39246e-005 ...
9.04675e-006 6.7236e-005 3.79383e-011 4.5712e-006 9.9447e-005 1.04075e-005 8.61333e-005 7.15437e-005 8.87927e-006 ...
9.91392e-005 1.24858e-005 1.42515e-005 5.49853e-006 4.00286e-006 2.63873e-005 7.32161e-006 5.25322e-005 1.43838e-005 ...
6.6506e-005 4.2387e-005 1.21336e-006 5.51425e-005 1.90663e-005 8.49555e-006 1.30891e-005 2.89695e-006 5.31321e-006 ...
1.73879e-005 1.52146e-005 4.75415e-006 9.47949e-005 3.64846e-007 6.03109e-006 0.000181987 6.85426e-006 0.000162451 ...
9.27616e-005 4.90602e-006 0.000377911 1.22856e-005 1.10826e-005 2.38599e-005 8.54074e-006 2.80267e-006 2.42106e-005 ];
err_qmoy50=[...
1.25685e-005 9.68019e-006 6.70498e-006 5.34067e-006 1.80397e-006 9.48663e-006 0.000139122 9.64947e-006 5.12539e-005 ...
1.78365e-005 1.1428e-006 3.31421e-005 0.000139122 9.64947e-006 5.12539e-005 1.78365e-005 1.1428e-006 3.31421e-005 ...
7.75568e-006 5.35703e-005 1.07022e-005 9.85725e-006 1.73022e-006 2.96718e-005 6.39114e-005 8.4084e-006 2.81318e-006 ...
3.75904e-005 3.90468e-006 6.75595e-006 2.17815e-006 8.36218e-006 9.04714e-006 3.47216e-006 3.49444e-006 1.27678e-005 ...
3.61841e-006 8.07571e-006 1.26081e-005 5.49951e-006 1.96392e-006 2.16315e-005 1.13417e-006 4.10199e-006 8.34388e-005 ...
8.15516e-007 1.08669e-006 0.000138451 8.65069e-006 6.74189e-005 2.78547e-006 6.06964e-006 2.55973e-005 7.5076e-005 ...
7.93699e-006 4.03693e-005 1.25439e-011 4.56683e-006 7.57907e-005 9.31281e-006 3.39925e-005 5.87142e-005 8.38879e-006 ...
7.6368e-005 8.73909e-006 1.16751e-005 5.60323e-006 2.74253e-006 1.53627e-005 2.94672e-006 5.11282e-005 1.10637e-005 ...
2.88522e-005 3.51019e-005 8.85415e-007 2.87861e-005 8.97493e-007 7.42807e-006 3.9965e-006 2.33543e-006 3.28428e-006 ...
8.81082e-006 1.62157e-006 5.93082e-006 5.24008e-005 1.71925e-008 3.98347e-006 0.00017861 2.13347e-006 6.83152e-005 ...
3.48443e-006 2.22702e-006 0.000163002 2.48089e-006 8.23389e-006 1.36445e-005 3.16451e-006 7.8893e-009 2.46532e-005 ];
err_qmoy40=[...
1.63744e-006 1.18553e-005 1.08451e-006 1.60798e-006 1.07155e-006 4.48426e-006 2.71642e-005 6.0546e-006 5.14246e-005 ...
1.57109e-005 2.38879e-007 2.64024e-005 2.71642e-005 6.0546e-006 5.14246e-005 1.57109e-005 2.38879e-007 2.64024e-005 ...
2.44343e-006 4.85749e-005 8.99379e-006 2.75832e-006 1.61005e-006 1.99259e-005 3.86037e-005 1.50048e-005 2.14003e-006 ...
3.68228e-005 3.57421e-006 2.39521e-006 5.66724e-006 5.6947e-006 2.42236e-006 1.13722e-006 3.45265e-006 9.93319e-006 ...
3.70532e-007 7.21146e-006 5.65491e-006 1.93335e-006 1.59254e-006 1.92847e-005 2.49283e-012 1.87268e-006 8.10842e-005 ...
1.17002e-006 6.88103e-006 6.68396e-005 8.9397e-006 6.55007e-005 5.71768e-008 7.25843e-011 1.8409e-005 2.82349e-005 ...
1.03553e-006 2.98016e-005 2.80267e-012 4.56682e-006 7.42393e-005 6.98006e-006 3.45069e-005 4.45303e-006 8.62847e-006 ...
7.34821e-005 8.28211e-006 1.00882e-005 5.16946e-006 1.51397e-008 5.5403e-006 5.92447e-006 5.08998e-005 6.99966e-006 ...
2.79804e-005 3.43301e-005 1.25712e-009 1.93502e-005 1.01094e-006 7.87964e-006 3.37974e-006 2.33529e-006 4.59375e-006 ...
5.56247e-006 3.35932e-006 6.04977e-006 5.271e-005 8.60167e-007 4.23368e-006 7.35272e-005 3.08513e-006 6.66935e-005 ...
2.50959e-006 6.33372e-006 0.00016135 3.12431e-006 1.00803e-005 1.40554e-005 1.70299e-006 3.45182e-010 2.44946e-005 ];
err_qmoy30=[...
1.17368e-005 1.20705e-005 1.94614e-007 3.97261e-006 1.07155e-006 3.87882e-006 2.48773e-005 3.08357e-006 3.76012e-005 ...
1.51265e-005 3.37872e-007 2.92317e-005 2.48773e-005 3.08357e-006 3.76012e-005 1.51265e-005 3.37872e-007 2.92317e-005 ...
1.06603e-006 4.99301e-005 7.687e-006 3.55152e-006 1.26201e-006 2.06705e-005 3.93334e-006 1.55787e-005 1.20709e-006 ...
2.53955e-006 2.24583e-006 1.02252e-006 1.10751e-012 4.55513e-006 6.60652e-007 8.0061e-007 4.65723e-006 6.32546e-006 ...
1.27426e-007 1.44932e-006 4.4658e-006 4.47858e-008 2.35924e-007 1.55759e-005 4.74443e-013 1.98959e-006 4.80863e-005 ...
```

1.26604e-012 6.29894e-006 6.5648e-005 1.03383e-005 3.89414e-005 8.80121e-013 1.08634e-006 1.73745e-005 1.96863e-005 ...
6.69465e-009 2.39762e-005 1.8914e-025 4.56682e-006 7.16674e-005 6.37944e-006 2.27788e-005 6.92755e-006 6.37671e-006 ...
7.31449e-005 7.52924e-006 6.17893e-006 7.74957e-009 7.03663e-009 4.35609e-006 4.75768e-008 4.77416e-007 2.59115e-006 ...
7.81701e-006 1.14087e-006 2.65612e-011 1.33714e-005 1.01072e-006 6.77502e-006 1.16732e-006 2.33531e-006 3.30102e-006 ...
4.90866e-006 1.58359e-006 5.42199e-006 3.32149e-005 1.90298e-006 7.23298e-011 6.34426e-005 9.2687e-007 2.47692e-005 ...
5.96211e-007 2.96966e-006 4.98911e-005 1.89774e-006 7.07077e-006 4.6071e-006 7.8698e-007 2.91772e-011 1.64447e-005 ];
err_qmoy20=[...
2.30759e-006 1.65645e-006 1.28368e-007 5.4661e-006 9.95373e-007 1.42205e-007 1.42181e-007 1.66562e-006 3.2593e-005 ...
2.84028e-007 1.64319e-011 2.85388e-005 1.42181e-007 1.66562e-006 3.2593e-005 2.84028e-007 1.64319e-011 2.85388e-005 ...
5.2032e-007 5.13119e-005 8.50074e-006 2.28803e-006 1.26148e-006 1.85234e-005 1.47134e-008 1.58632e-006 7.82293e-008 ...
1.34168e-008 8.11396e-007 1.56027e-006 1.20877e-008 3.78862e-006 1.38303e-007 8.66669e-020 3.04685e-011 5.53139e-006 ...
5.80422e-007 3.70559e-006 4.32173e-006 8.48618e-007 1.94159e-011 8.32263e-006 0 1.15526e-006 4.80759e-005 ...
0 3.12954e-006 6.49385e-005 9.84677e-006 3.88411e-005 0 8.36688e-011 1.41255e-005 1.02649e-005 ...
4.49189e-011 1.20547e-005 8.89302e-026 1.27789e-009 6.12662e-005 6.2956e-007 1.18233e-005 2.23406e-007 8.73888e-007 ...
6.22848e-005 4.95439e-006 6.6271e-006 3.58404e-013 3.42372e-009 3.30005e-006 6.38714e-013 5.4842e-013 2.2975e-006 ...
2.23762e-007 1.75137e-008 2.61696e-011 3.56095e-006 7.21727e-007 7.4985e-006 1.2218e-006 2.3402e-006 2.7908e-006 ...
3.72335e-006 1.00069e-012 1.08793e-006 2.11427e-005 0 7.53115e-011 5.75954e-005 4.69807e-011 1.57613e-005 ...
0 7.8405e-011 3.72642e-005 0 1.08399e-007 3.08211e-006 0 2.86422e-011 5.94159e-006 ];
err_qmoy10=[...
3.49713e-009 7.16438e-008 7.95984e-010 7.21354e-007 9.95373e-007 1.62018e-007 2.3071e-010 1.33219e-007 1.97515e-005 ...
6.11404e-013 1.42537e-011 1.49999e-005 2.3071e-010 1.33219e-007 1.97515e-005 6.11404e-013 1.42537e-011 1.49999e-005 ...
4.59841e-008 4.65637e-005 4.47932e-007 2.29038e-006 1.10633e-006 4.50413e-006 2.42688e-009 4.2716e-011 1.85404e-008 ...
1.43838e-009 3.96506e-011 2.28866e-008 1.20877e-008 4.89281e-007 2.34431e-008 5.47983e-020 2.77408e-011 1.66542e-006 ...
0 3.05628e-007 2.06194e-006 0 1.8063e-011 3.86276e-006 0 4.84344e-007 8.40579e-007 ...
0 8.05915e-011 8.01313e-006 4.35844e-006 7.48636e-008 0 8.80289e-011 7.20284e-006 1.65011e-012 ...
4.68734e-011 3.27374e-006 8.89302e-026 8.56546e-011 2.0957e-005 2.18102e-008 6.39492e-006 6.84974e-007 1.95003e-006 ...
2.39443e-005 5.47611e-007 3.54162e-006 0 3.0788e-011 2.26016e-006 0 0 2.82935e-007 ...
2.64127e-008 3.64023e-013 2.36267e-011 2.03929e-007 7.21735e-007 3.44992e-006 1.22183e-006 2.34028e-006 4.3246e-007 ...
1.66081e-007 0 1.08793e-006 9.15947e-006 0 7.51624e-011 1.07831e-005 4.16532e-011 4.07068e-006 ...
0 7.94626e-011 1.34239e-005 0 4.08039e-011 9.04741e-007 0 2.65482e-011 1.77837e-006 ];

```
% calcul de l'intervalle de confiance selon la distribution de Student
err_QMoy_C=[Stu_Const*std(err_qmoy95)/sqrt(length(err_qmoy95))...
        Stu_Const*std(err_qmoy90)/sqrt(length(err_qmoy90))...
        Stu_Const*std(err_qmoy80)/sqrt(length(err_qmoy80))...
        Stu_Const*std(err_qmoy70)/sqrt(length(err_qmoy70))...
        Stu_Const*std(err_qmoy60)/sqrt(length(err_qmoy60))...
        Stu_Const*std(err_qmoy50)/sqrt(length(err_qmoy50))...
        Stu_Const*std(err_qmoy40)/sqrt(length(err_qmoy40))...
        Stu_Const*std(err_qmoy30)/sqrt(length(err_qmoy30))...
        Stu_Const*std(err_qmoy20)/sqrt(length(err_qmoy20))...
        Stu_Const*std(err_qmoy10)/sqrt(length(err_qmoy10))];
% calcul des moyennes
moy_QMoy_C=[mean(err_qmoy95) mean(err_qmoy90) mean(err_qmoy80)...
mean(err_qmoy70) mean(err_qmoy60) mean(err_qmoy50) mean(err_qmoy40)...
mean(err_qmoy30) mean(err_qmoy20) mean(err_qmoy10) ];%calcul des moyennes
```